

La pratique du TRS-80

Volume I - Architecture et Basic II

**Pierre Giraud
Alain Pinaud**

La pratique du TRS-80

Volume I - Architecture et Basic II

Dans la même collection

Programmer en BASIC — Michel Plouin
Programmer en L.S.E. — Stéphane Berche et Yves Noyelle
La découverte du P.E.T. — Daniel-Jean David
La découverte de l'Apple II — Frédéric Lévy et Dominique Schraen

Tous droits de traduction, d'adaptation et de reproduction par tous procédés
réservés pour tous pays

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les «copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective» et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, «toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite» (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

© Editions du P.S.I. 9, rue d'Orgemont 77400 LAGNY/MARNE - 1979
ISBN : 2-86470-005-0

imprimé en France

La pratique du TRS-80

Volume I - Architecture et Basic II

par

Pierre Giraud et Alain Pinaud



Editions du P.S.I.

1981

Les auteurs de cet ouvrage sont informaticiens chez le constructeur français d'ordinateurs CII HONEYWELL BULL, où ils exercent leur activité depuis une quinzaine d'années ; ce qui leur a permis de suivre les mutations prodigieuses des unités centrales d'ordinateurs passant de la taille d'une armoire à celle d'un morceau de sucre.

Tous deux, ingénieurs d'études :

Pierre Giraud, 36 ans, a abordé l'informatique par le matériel au début de sa carrière, mais est vite venu au logiciel, où il a participé à l'étude et à la réalisation de logiciels de base et d'applications temps réels, saisie de données, langages.

Alain Pinaud, 34 ans, participe à l'étude et la réalisation de systèmes spéciaux, tant sur le plan matériel que logiciel.

Leur expérience des petites machines a fait qu'avec l'apparition sur le marché des petits systèmes individuels et pour se changer les idées, en dehors de leurs activités professionnelles, ils pratiquent le TRS-80... !

P R E S E N T A T I O N

Vous avez maintenant en mains le premier volume d'une série consacrée au TRS-80. Nous avons en effet éprouvé le besoin de mettre par écrit une somme d'expériences personnelles et de renseignements tirés de lectures - Afin de mieux vous faire connaître, ou de vous apprendre à mieux utiliser cette petite merveille qu'est le TRS-80, ainsi que de vous faire bénéficier de "trucs" et astuces qui nous ont coûté un certain nombre de nuits blanches... et dont il serait inutile que vous fassiez les frais.

Cette série de volumes s'adresse à une catégorie de lecteurs déjà initiés aux mystères de l'informatique individuelle et possédant de préférence, un TRS-80, bien que le présent volume fasse lui-même figure d'initiation.

Revenons au volume 1 : C'est le volume d'introduction de la série. Il vous apportera les bases techniques (technologie et programmation) permettant d'aborder avec sérénité les volumes suivants, consacrés à une étude plus approfondie sur le fonctionnement du TRS-80 (et de ce qui l'entoure), son dépannage éventuel (sa maintenance), ses améliorations possibles et son exploitation.

Mais, il y a toujours un dilemme lorsque l'on écrit ce genre de livre, il est trop ardu pour certains, pas assez pour d'autres. Voici notre réponse : (mais nous devons tout d'abord réclamer votre clémence), vous savez, quinze ans de pratique de l'"Informatique", ça use !... ça use !... et il arrive, parfois, par habitude, de comprendre facilement des choses difficiles (pour certains), d'expliquer difficilement des choses faciles (pour d'autres)... Vous voyez, Docteur, ça recommence ! Nous comptons donc, amis Lecteurs, sur vos critiques et suggestions.

Nous avons essayé, dans ce volume, d'adopter pour démarche d'expliquer chaque terme nouvellement introduit, dans la mesure où celui-ci ne figure pas forcément dans les ouvrages d'initiation. Si ces définitions sont évidentes pour vous, sautez-les, et si cet ouvrage ne vous apporte rien de nouveau, ce qui est très possible, il vous fournira une base documentaire à laquelle vous pourrez vous référer en cas de besoin, une somme d'informations condensées dans un ultime but : dominer Thérèse...

Ah oui ! au fait, nous ne vous avons pas présenté : nos épouses, n'ayant pas toujours apprécié nos nuits passées avec le TRS, l'ont surnommé : Thérèse !

S O M M A I R E

1 - INTRODUCTION	Pages
. Le besoin ou la nécessité	9
. Pourquoi le TRS-80 ?	11
. Petit lexique préliminaire	13
 2 - PRESENTATION GENERALE	
. Le matériel	15
. Le logiciel de base et les utilitaires	18
. A qui s'adresse-t'il ?	20
. Quelques applications réalisées	22
 3 - APPROCHES TECHNIQUES	
. Le matériel	27
. Le logiciel	35
 4 - LE BASIC II	
. Introduction pratique	41
. Commandes moniteur	46
. La commande d'édition	50
. Les symboles de base	52
. Les constantes et les variables	53
. L'instruction d'affectation	57
. Evaluation des expressions arithmétiques	58
. Les fonctions standards	60
. Les instructions de saut et de test	66
. Les boucles	73
. Les instructions d'Entrée/Sortie	78
. Les fonctions graphiques	90
. Les fonctions spéciales	91
 5 - EXEMPLES DE PROGRAMMES	
. Conjugaison	95
. "Devine ?"	96
. Saisie de données	99
 A N N E X E S	
 I Caractéristiques du BASIC II	105
II Les codes erreurs du BASIC II	106
III Mots réservés du BASIC II	107
IV Codes clavier et graphiques	109
V Problèmes de magnétophone	113
VI Le BASIC niveau I	116
VII Conversions décimales - hexadécimales	121

CHAPITRE 1

INTRODUCTION

Le besoin, ou la nécessité d'un ordinateur personnel ?

Pourquoi ? Pour qui ? Pour réaliser l'un des vieux rêves de l'homme, en voulant donner un semblant de vie et un semblant d'intelligence à ce qui n'est jamais qu'un "tas de ferraille inerte" ? Un besoin impérieux de dominer la machine (mais en lui fournissant les armes pour se défendre) ? Un support pour étancher sa soif de création, ses projections ?

Oh ! nous n'allons pas essayer ici d'inventer des prétextes ou des histoires à dormir debout.

La vérité, elle est toute simple : nous faisons partie des bienheureux qui ont la chance d'aimer le travail qu'ils font, et de l'aimer avec acharnement, à tel point que, non contents de manipuler des ordinateurs toute la journée, ils veulent aussi pouvoir continuer chez eux, le soir, une partie des week-ends ou de la nuit même.

Laissons-là la philosophie ; et puis, pourquoi faudrait-il toujours trouver des réponses à tout ! Nous ne sommes pas des ordinateurs !

Pensez donc ! En une seule soirée, vous pouvez quitter New-York en voilier, rejoindre San-Francisco par le Cap Horn, contre vents et marées, faire un golf de dix-huit trous, battre Thérèse aux échecs, débarrasser la galaxie Tartampion des infâmes monstres de l'espace qui menaçaient de s'en emparer, apprendre votre

vocabulaire d'anglais quotidien, interroger votre fichier de comptes, sortir un listing des factures impayées, vous apercevoir que votre femme a dépensé pour 1256,72 francs de chaussures durant le dernier trimestre, vous faire battre par Othello, perdre 12500 dollars au Blackjack (quel tricheur celui-la !), vous faire battre de nouveau par Othello, vous transformer en pilote de chasse se frayant un chemin au milieu des salves... Boum !... Boum!... (Là, ce sont les voisins qui frappent au mur...), et tout cela, sans le moindre signe de fatigue ou de contrariété (sauf pour les chaussures, là, elle y a été un peu fort !).

En attendant la note que l'ordinateur décernera à votre fils, en réponse aux multiplications qu'il est en train de lui poser, pensez aux autres idées pratiques que vous pourrez développer demain sur votre joujou, si vous pouvez toutefois en avoir l'accès, car votre femme qui passe son examen de dactylo dans trois jours, voudrait bien avoir la certitude qu'elle "tient" toujours la vitesse de 56 mots/minute, soit 585 caractères en une minute et quarante trois secondes, dixit Thérèse...

Bien sûr ! un micro-ordinateur, on peut s'en passer ! mais c'est tellement plus compliqué avec... Naturellement, il y a des applications beaucoup plus sérieuses, mais patience.

POURQUOI LE TRS-80 ?

Il y aura inévitablement dans ce qui suit un certain nombre de termes techniques, mais ne vous y arrêtez pas trop, ils seront expliqués en temps utile. Quant aux répétitions, elles sont là pour vous aider à mieux mémoriser toutes ces choses nouvelles pour vous.

Cinq facteurs principaux ont orienté notre choix vers le TRS-80, sans parler du fait qu'à l'époque de son achat, il n'avait pas tous les frères qu'il a maintenant, tout au moins en France, (mais cela ne remet toujours pas notre choix en cause) :

- *Son prix* : Défiant alors, et encore actuellement, toute concurrence, (à partir de 3995 FF en mars 79, tout compris, en version level 1, avec un clavier standard permettant la dactylographie, un écran vidéo, un bloc d'alimentation et un magnétophone).

- *Son microprocesseur* : Z-80, (le cerveau du système), de la firme américaine Zilog. En tant que professionnels, nous avons bien entendu apprécié ce choix excellent, d'autant que le Z-80 nous était déjà connu par son cousin, le 8080 de la firme Intel. Ce microprocesseur possède les qualités, - vitesse, richesse du jeu d'instructions (son vocabulaire), facilité d'emploi, grande souplesse - qui le font retenir dans la majorité des applications industrielles sophistiquées à caractère professionnel (par opposition aux applications de type "grand public") nécessitant fiabilité et performance. L'intérêt que nous voyons également dans ce microprocesseur réside dans la diffusion universelle de son langage (incluant celui du 8080), ce qui sous-entend une "bibliothèque" de sous-programmes de taille respectable. Par exemple, si vous voulez faire résoudre tel ou tel problème à votre ordinateur, il y a de fortes chances pour que ce problème soit déjà traité par un "sous-programme" de la bibliothèque des utilisateurs, alors inutile de vous casser la tête.

- *Ses possibilités d'extension* : Entendons par là : le moyen de communiquer facilement avec des appareils extérieurs à l'ordinateur. Sur ce plan, le TRS-80 est bien placé, puisqu'il présente, par le moyen d'une prise arrière, quarante fils ou signaux aux utilisations les plus diverses. Notons qu'à l'inverse de certains micro-ordinateurs, qui offrent des moyens de communication figés - et donc forcément limités -

le TRS-80 donne généreusement la quasi-totalité du "Bus" (ou canal de communication) du microprocesseur lui-même. De ce fait, tout ce qui est propre à se raccorder au microprocesseur Z-80 (ou autre, ayant une structure de Bus similaire), pourra se raccorder au TRS-80. Voilà ce que nous appelons un système "ouvert".

- *Les performances de son logiciel* : Le logiciel est le terme qui désigne l'ensemble des programmes qui vont donner la "matière grise" indispensable à la machine.

En parlant ici de performances, nous nous empressons de dire que nous pensons en particulier au logiciel "Level 2" du TRS-80 qui, moyennant moins de 700 FF supplémentaires, pourra être greffé sur la machine de base dont le prix est cité plus haut.

Mais ce "changement de personnalité" en vaut la peine. Le logiciel "Level 1" est, en effet, destiné à l'initiation et s'avère, de ce fait, très limité lorsque son but est atteint. On éprouve alors rapidement le besoin de passer au stade supérieur.

Le "Level 2", par contre, écrit par des spécialistes du langage d'ordinateur, offre une richesse et une souplesse qui feraient pâlir des ordinateurs bien plus puissants.

Comme pour le matériel, on retrouve ici, dans le logiciel, la notion de système ouvert qui permet, avec une grande facilité pour l'utilisateur, d'agir directement sur les différents organes de la machine.

- *Sa logeabilité et son autonomie* : Ce dernier facteur peut sembler un détail, mais il a, lui aussi, son importance. N'ayant pas l'audace d'aller jusqu'à transformer notre salle de séjour en laboratoire d'étude ou en centre de calcul, (...d'accord pour Thérèse, mais une Thérèse discrète...), il fallait, bien entendu, trouver une machine qui s'intègre facilement dans un mobilier donné, en l'occurrence l'un des secrétaires d'un meuble-living, le clavier du TRS étant fixé sur l'abattant, le vidéo et le magnétophone eux, rentrant facilement au fond du secrétaire, laissant même la place à de futures extensions. Ce P.S.I. (petit système individuel) a également l'avantage de vous permettre une autonomie vis-à-vis du téléviseur familial, bien appréciable lorsque madame a décidé de regarder Paul Newman à la télé, ce soir (...d'accord pour Thérèse, mais pas pour une Thérèse envahissante...).

Puisqu'il est question d'écran, il faut préciser que celui-ci, lorsqu'il est bien réglé, (ce qui n'est malheureusement pas toujours le cas), offre une grande lisibilité de seize lignes nettement séparées, de soixante-quatre caractères chacune, avec possibilité, en Level 2, de gros caractères très agréables.

PETIT LEXIQUE PRELIMINAIRE

Avant d'aller plus loin dans notre étude, il convient maintenant d'examiner quelques termes de base qui seront utilisés fréquemment au cours des chapitres suivants. Il n'y a pas forcément de liens logiques entre ces termes, aussi n'en cherchez pas...

Moniteur Vidéo : C'est, en somme, un téléviseur, mais privé de la partie réception des ondes donnant le son et l'image, car inutilisée dans notre cas. Il se compose globalement des sous-ensembles suivants : - amplificateur vidéo, c'est un circuit électronique qui amplifie les signaux électriques venant de l'extérieur (clavier du TRS-80, par exemple) avant de les diriger vers l'écran ; - des circuits de "balayage" qui permettent de déplacer un point lumineux (appelé spot), très rapidement de gauche à droite, et de haut en bas, ce qui donne, grâce à la persistance rétinienne de l'oeil, l'impression d'une image fixe ; - des circuits d'alimentation (voir ce mot plus loin) qui fournissent, à partir du secteur, les différentes tensions nécessaires ; - un tube cathodique (écran), qui donne l'image.

Alimentation : C'est un dispositif permettant de fournir la ou les tensions électriques nécessaires aux circuits électroniques de l'ordinateur. Une pile, par exemple, est une alimentation simple. Mais il y a une difficulté : les circuits annexés au microprocesseur utilisent fréquemment trois tensions différentes (+5V, -5V, +12V). Le bloc d'alimentation aura donc pour rôle de fournir, à partir de la tension secteur 220V, les tensions nécessaires.

Clavier : Tout le monde a déjà vu un clavier de machine à écrire. Le TRS-80 en comporte un également, mais sans toute cette mécanique ! C'est l'un des organes d'entrée qui permet d'envoyer des informations au microprocesseur. Chaque touche du clavier abrite, en effet, un petit contact qui, lorsqu'il se ferme, indique au microprocesseur la touche correspondante qui a été enfoncée.

Magnétophone : Tout le monde le connaît aussi, mais à quoi peut-il bien servir dans ce cas ? Il s'agit, ici, d'un magnétophone à cassettes tout-à-fait ordinaire, mais qui a un autre but que de faire de la musique ! Lorsque vous voulez vous souvenir de quelque chose, vous l'écrivez sur un papier ? En relisant ce

papier, plus tard, vous vous souvenez alors de ce que vous y aviez écrit.

L'ordinateur agit de même avec le magnétophone. Lorsque vous voulez qu'il se souvienne d'un jeu, par exemple, vous devez lui donner l'ordre d'aller "l'écrire" sur une cassette. Plus tard, après l'avoir privé un court instant de sa source d'alimentation, il faudra lui donner l'ordre de "relire" la cassette, afin qu'il se re-souviene de ce jeu. Le rôle du magnétophone est donc celui de support de mémorisation.

Programme : C'est un ensemble d'instructions ou directives exploitées par l'ordinateur, pour réaliser une tâche donnée. A chaque tâche, que ce soit un jeu, un calcul de paie ou une opération quelconque, correspond un programme spécifique à cette tâche.

Le programme est placé dans la mémoire de l'ordinateur où il est exécuté au fur et à mesure.

Par opposition à l'ensemble des composants électroniques nommé "matériel", le programme (ou l'ensemble des programmes) est nommé "logiciel" (les mots américains correspondants sont "hardware" et "software").

Basic : C'est un langage évolué universel, c'est aussi un programme. Mais il n'a pas besoin, dans le cas du TRS-80, d'être chargé en mémoire, étant donné qu'il est déjà dans une mémoire spéciale qui ne s'efface jamais (pratiquement), et qui est opérationnelle dès la mise sous tension du système.

Sans programme, l'ordinateur est mort. C'est pour cela qu'il doit toujours y en avoir un en mémoire centrale, si petit soit-il, afin de diriger les opérations initiales à la mise en marche de l'appareil. Ce petit programme est appelé "Bootstrap" en américain, ce qui signifie à peu près "chausse-pied". Son rôle sera uniquement d'assurer l'entrée en mémoire d'un programme extérieur plus important qui est, en fait, le programme principal. Mais dans le cas du TRS-80, ce dispositif est superflu puisque, rappelons-le, la totalité du Basic est toujours présente en mémoire.

CHAPITRE 2

PRESENTATION GENERALE DU TRS 80

Le matériel

En version de base, le TRS-80 se compose globalement de quatre "boîtes" indépendantes :

Un ensemble clavier : Il abrite également la carte (circuit imprimé) micro-ordinateur (microprocesseur, mémoires, circuits de couplage vidéo et magnétophone). Cet ensemble est muni de 3 prises d'entrée de type DIN 5 points, d'un interrupteur marche/arrêt, d'un bouton-poussoir de réinitialisation, d'un voyant de mise sous tension et d'un clavier au standard anglo-saxon (QWERTY) de 53 touches.

A l'arrière du clavier, on trouve également un connecteur 40 points, permettant la liaison avec d'éventuelles extensions. Les trois prises DIN sont utilisées pour raccorder respectivement : le bloc alimentation, le moniteur vidéo et le magnétophone. Le logiciel de base (Basic 1 ou 2), ainsi que la mémoire RAM (4K ou 16 K) sont logés dans cet ensemble.

Le moniteur vidéo : Il ressemble à un petit récepteur de télévision portable, et comporte un cordon d'alimentation secteur, un interrupteur marche/arrêt, un cordon vidéo se branchant sur l'ensemble clavier et deux commandes de réglage de luminosité et de contraste. Des commandes plus "techniques" sont accessibles à l'arrière de l'appareil. L'écran vidéo permet l'affichage de 16 lignes de 64 caractères alphanumériques (ou 16 lignes de 32 caractères de taille double) ainsi que d'une grille graphique de 128 fois 48 cellules.

Le magnétophone : C'est un magnétophone à cassettes standard, mais possédant une télécommande du moteur (commande à distance), une entrée écriture cassette (AUX) et une sortie lecture (EAR). Un câble, terminé à l'une de ses extrémités par trois prises "Jack" permet de relier ces trois fonctions à l'ensemble clavier (prise DIN 5 points).

Bloc alimentation : C'est une petite boîte noire, comportant un cordon secteur 220V et un cordon terminé par une prise DIN s'enfichant sur l'ensemble clavier. Cette alimentation fournit les tensions nécessaires au fonctionnement du TRS-80. Attention au croisement malencontreux des trois prises DIN du clavier... Il pourrait se traduire par une légère fumée... (voir les repères à l'arrière du clavier).

Un certain nombre d'extensions venant s'ajouter à l'ensemble de base sont proposées par le constructeur :

Extension mémoire RAM 16 K octets : Cette mémoire occupe physiquement la place de la mémoire de base de 4 K octets située sous le clavier du TRS-80. Les 8 boîtiers mémoire 4 K sont retirés et remplacés par 8 boîtiers 16 K.

Extension du logiciel : Le Basic 1 contenu dans 2 mémoires ROM de 2 K octets chacune, est remplacé par le Basic 2 contenu dans 3 mémoires ROM de 4 K octets chacune. Pour ce faire un circuit imprimé annexe contenant ces ROM est ajouté sous le clavier et est relié à la carte mère par un câble plat.

L'expansion interface : Cet ensemble, de dimensions à peu près égales à celles de l'ensemble clavier, se situe, en principe, sous le moniteur vidéo. Il est relié au clavier par un câble plat. Les possibilités de l'expansion interface sont les suivantes :

- extension mémoire RAM portée à 48 K octets (32 K dans l'expansion et 16 dans le clavier),
- couplage de 1 à 4 disques souples (mini-disquette),
- couplage d'une imprimante parallèle,
- contrôle d'une horloge,
- couplage d'un deuxième magnétophone à cassettes.

De plus, il est possible d'ajouter dans le châssis une carte aux normes RS-232 (V 24 en Europe), permettant de raccorder

un terminal clavier/écran ou une imprimante série, ainsi qu'une liaison par ligne téléphonique.

Bien entendu, le connecteur de sortie de l'ensemble clavier est déporté sur l'expansion interface, afin de pouvoir raccorder d'autres extensions.

Les périphériques : Le constructeur commercialise : imprimantes (deux modèles), disquettes (5 pouces, 89600 octets par disquette) modem acoustique et synthétiseur vocal.

D'autres sociétés diffusent, souvent à des prix compétitifs, des extensions RAM 16 K (utilisables aussi dans l'expansion interface) les périphériques standards imprimantes et disquettes, et des dispositifs originaux, non commercialisés par le constructeur.

LE LOGICIEL DE BASE ET LES UTILITAIRES

Le logiciel de base comprend le ou les programmes qui contrôlent la machine, ceux qui ont la part la plus active à son fonctionnement et à son environnement (périphériques). Les utilitaires, ainsi nommés parce qu'ils réalisent un tas de petites choses utiles, sont toujours des programmes "extérieurs" contenus sur cassettes ou disquettes.

Le logiciel de base fourni par le constructeur comprend l'interpréteur Basic (2 versions, 2 puissances) et le TRSDOS (DOS = Disk Operating System).

Le Basic, contenu en ROM, est opérationnel dès la mise sous tension de la machine. C'est un programme qui permet au TRS de parler un langage du même nom, et connu aussi d'autres ordinateurs. Il a été choisi pour sa facilité d'emploi, ses possibilités variées et son apprentissage facile pour l'homme. Comme son nom l'indique, il interprète et exécute les programmes écrits en langage Basic.

Le TRSDOS : Disponible uniquement lorsqu'une disquette est raccordée au TRS, est un dictateur qui contrôle toute la machine. Avec lui, même l'interpréteur Basic devient esclave, bien que le TRSDOS donne à ce dernier une puissance encore plus grande qu'auparavant. Tous les programmes sont stockés sur disque, d'où ils sont rappelés en quelques dixièmes de secondes. D'autres langages peuvent aussi être soumis au TRSDOS (FORTRAN, par exemple).

Les utilitaires fournis par le constructeur sont au nombre de deux (sous Basic 1 ou 2) : TBUG et EDTASM.

TBUG est un petit "moniteur" permettant d'écrire des programmes dans le langage directement assimilable par le microprocesseur (hexadécimal).

EDTASM est un "éditeur assembleur" permettant d'écrire des programmes en langage d'assemblage (très proche du langage machine).

D'autres sociétés proposent enfin tout un choix de logiciels que nous nous contenterons d'énumérer, et dont certains seront décrits ultérieurement :

- compilateur FORTRAN sous TRSDOS
- macro-assembleur sous TRSDOS
- un autre DOS : CP/M
- compilateur Basic sous CP/M (CBasic)
- langage LISP
- COBOL sous CP/M
- RSM (moniteur désassembleur)
- RENUM (utilitaire de renumérotation Basic)
- DUPLIK (utilitaire de duplication de cassettes)
- NEWDOS (donne au TRSDOS des fonctionnalités supplémentaires tout en corrigeant certaines anomalies).
- tout récemment : langage PASCAL sous TRSDOS.

A QUI S'ADRESSE T-IL ?

Sans vouloir reprendre une formule connue (de sept à soixante dix sept ans), on peut dire qu'il peut commencer à intéresser et à être utile aux enfants de dix à douze ans (surdoués mis à part).

Aux Etats-Unis, des spots publicitaires de la TV montrent chacun des membres de la famille devant son TRS-80 individuel, exerçant les activités les plus diverses. Bien entendu, nous n'en sommes pas là, mais qui aurait osé parler d'ordinateur individuel, il y a seulement cinq ans ? De nos jours, chaque enfant est capable d'utiliser un téléphone ou une calculatrice. Il y a vingt ans, ses parents ou grands-parents le pouvaient-ils ?

Il y a deux groupes principaux d'utilisateurs : *Ceux qui ne connaissent rien aux ordinateurs et qui ne veulent rien connaître de plus.* Ceux-là auront uniquement pour tâche de fournir à l'ordinateur un programme réalisé par les soins d'un programmeur ou d'une société de service en informatique ; *ceux qui connaissent ou qui ne connaissent pas, mais qui veulent en savoir plus, et notamment réaliser leurs propres programmes* (ou éventuellement modifier des programmes existants). Nous adressons nos félicitations à ceux de ce dernier groupe, car ils ont choisi la voie la plus passionnante. Bien sûr cette voie nécessitera un petit effort, mais c'est le premier pas qui coûte et il a déjà été franchi, puisque vous lisez ce livre.

Le TRS-80, de par sa conception, s'adapte facilement aux domaines d'application suivants :

- personnel

- . jeux, distractions, hobby
- . télécommandes diverses (train électrique par ex.)
- . gestion (finances familiales)
- . fichier (cuisine, téléphone, langues)
- . études personnelles (maths, langues, dactylo, musique)

- professionnel

- . fichier clients
- . gestion de stocks
- . banque de données
- . enseignement

- industriel

- . simulation
- . laboratoire
- . pilote d'automate
- . calcul scientifique
- . télécommandes industrielles
- . bancs de mise au point
- . mini système de développement logiciel

La liste ci-dessus n'est pas exhaustive et donnera simplement un échantillon des possibilités dont l'éventail s'élargit chaque jour. En fait, chaque sous-domaine d'application se décomposerait en de multiples branches. Mais le but recherché ici n'est que de vous donner une petite idée sur les possibilités d'adaptation de ce magicien qu'est l'ordinateur.

QUELQUES APPLICATIONS REALISEES

Nous avons choisi de vous présenter deux applications concrètes, parmi une foule d'autres probablement, et qui ont été portées à notre connaissance par leurs auteurs que nous tenons à remercier ici. Les pages qui suivent montrent que le TRS-80 est capable, quand il le veut, d'abandonner les jeux pour s'attaquer à des choses plus sérieuses.

Outre ces deux applications, nous aimerions citer au passage l'utilisation faite du TRS-80 à l'Université de Vincennes, en ce qui concerne l'étude des langages (la linguistique) et la transposition sur TRS-80 du langage LISP (VLISP), qui est un langage d'ordinateur orienté vers l'intelligence artificielle et la résolution de problèmes à caractères logiques. Thérèse sait maintenant parler le "LISP" !

Le traitement et l'affichage des résultats des élections cantonales.

Lors des dernières élections cantonales, un TRS-80 a été utilisé par un hebdomadaire régional, "Toutes les nouvelles de Versailles". Il s'agissait d'une opération aussi bien de stockage et de traitement des résultats que de "public address". Les résultats arrivant des différentes communes, souvent bureau de vote par bureau de vote, étaient collectés par des journalistes qui transmettaient immédiatement ces chiffres à l'un des leurs, spécialisé dans l'utilisation d'un ordinateur et, en l'occurrence, auteur du programme. Les chiffres, entrés par le clavier, étaient immédiatement traités par le TRS-80. Les totaux se faisaient sur des tableaux, commune par commune et canton par canton. Les pourcentages de participation et des voix des candidats étaient aussitôt calculés et affichés.

La grande originalité de cette opération a été de relier le TRS-80 à un réseau de télévision intérieur comportant une dizaine de postes, certains destinés au public venu en grand nombre s'enquérir du résultat de cette consultation, d'autres tenant informés les journalistes des résultats déjà reçus.

D'autres postes de télévision, disposés dans des salles de réception ou des bureaux, étaient destinés à la rédaction en chef et aux nombreuses personnalités présentes, qui avaient compris que ce procédé moderne et original leur permettrait d'être informés, minute après minute, de la physionomie de cette élection.

Sur les écrans apparaissaient le nom de la commune ou du canton, les inscrits, les votants, les suffrages exprimés, le nom de chaque candidat ainsi que le nombre de voix qu'il avait obtenu, tous les pourcentages arrondis au plus près, à deux chiffres après la virgule.

Enfin, l'ordinateur affichait, de lui-même : soit "résultats partiels", soit "ballotage", ou encore si l'un des candidats se trouvait élu, "ELU : M. DUPONT (UDF)", par exemple.

Précisons également que chaque résultat entré pouvait être facilement modifié, corrigé ou complété.

L'ordinateur individuel TRS-80 était un niveau 2 équipé de 16 K octets de RAM. Le programme Basic conçu pour cette opération et qui permettait également le passage d'un graphisme publicitaire pour le journal, pouvait s'exécuter sur seulement 5 K octets de mémoire.

Dans l'avenir, l'utilisation de MODEMS pour la transmission par téléphone, pourrait permettre les échanges des résultats avec d'autres hebdomadaires régionaux comme "Les Nouvelles de Rambouillet", par exemple.

Traitement de signal

Cette application, très technique, a été réalisée dans le domaine de l'électronique médicale mais, étant donné ses débouchés, nous l'examinerons sous un angle plus général.

Il s'agit d'extraire un signal noyé dans un bruit, en utilisant le principe de la détection synchrone.

- sortie de la courbe obtenue sur l'écran vidéo,
- sortie de la courbe sur table traçante,
- analyse de Fourier
- sortie du spectre d'amplitude et de phase sur table traçante.

La première étape consiste à définir, de façon simple et agréable, les paramètres d'acquisition du signal, d'où un mode conversationnel élaboré. Citons quelques paramètres :

- temps d'analyse du signal de 1 à 2000 MS.
- nombre d'échantillons : de 1 à 2048
- nombre de passages : de 1 à 65535.

La deuxième étape est l'acquisition proprement dite.

Le logiciel, écrit en assembleur et en Basic, utilise les paramètres ci-dessus et contrôle les fonctions suivantes :

- prise de N échantillons (T mini = 64 microsec.) et constitution d'un mot de 12 bits venant d'un convertisseur analogique numérique (CAN), et stockage dans une table tampon.
- cumul avec les échantillons précédents.
- bouclage

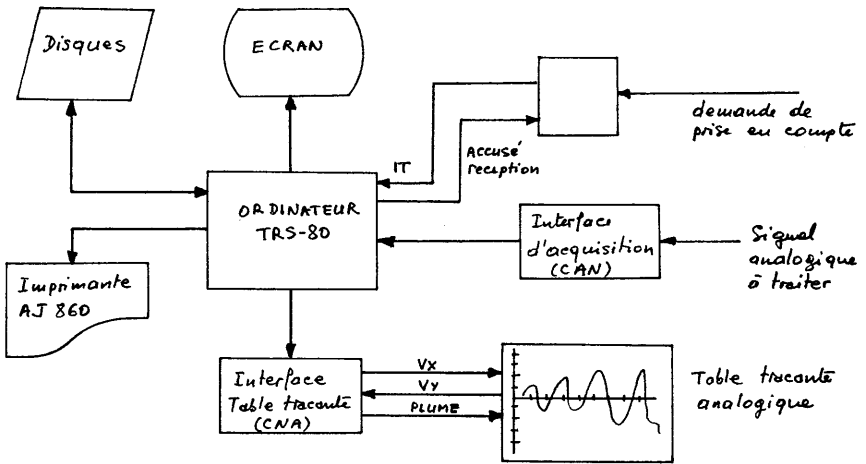
L'étape suivante consiste à visualiser sur l'écran la courbe obtenue et à la sortir sur une table traçante.

La carte d'interfaçage comprend deux convertisseurs numériques analogiques (CNA) de 12 bits (précision sur 200 mm : 0.05 mm) et une commande plume (position haute ou basse).

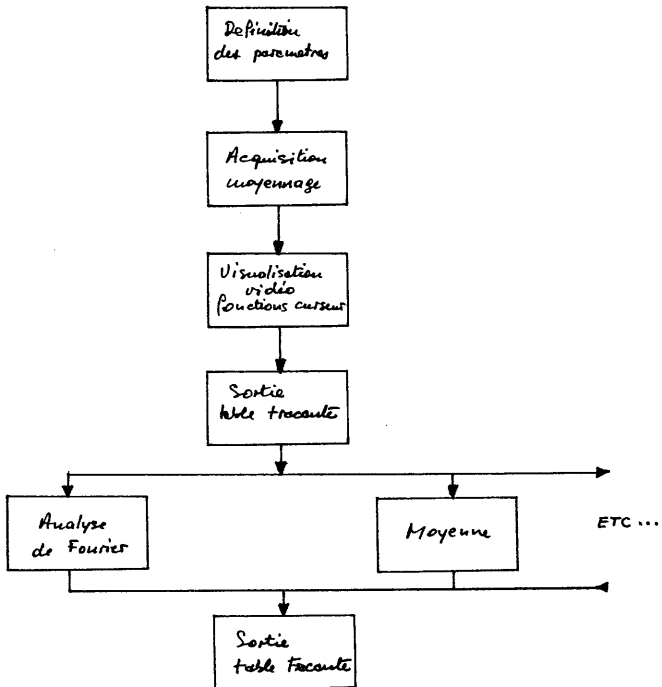
L'étape finale met ensuite en oeuvre un logiciel conversationnel permettant de définir les paramètres concernant le type de traitement à effectuer (analyse de Fourier, stockage des courbes sur disquettes, etc...)

Le schéma donne une idée de la configuration et des interfaces réalisés pour mener à bien cette application.

LA PRATIQUE DU TRS-80



SCHEMA FONCTIONNEL DU MATERIEL



SCHEMA FONCTIONNEL DU LOGICIEL

CHAPITRE 3

APPROCHES TECHNIQUES

Dans ce chapitre, nous allons "survoler" rapidement les bases matérielles et logicielles qui vont nous aider à examiner plus à fond l'anatomie du TRS-80.

Rappelons que ces bases peuvent être étayées par des ouvrages plus généraux consacrés à la technologie et à la programmation des ordinateurs.

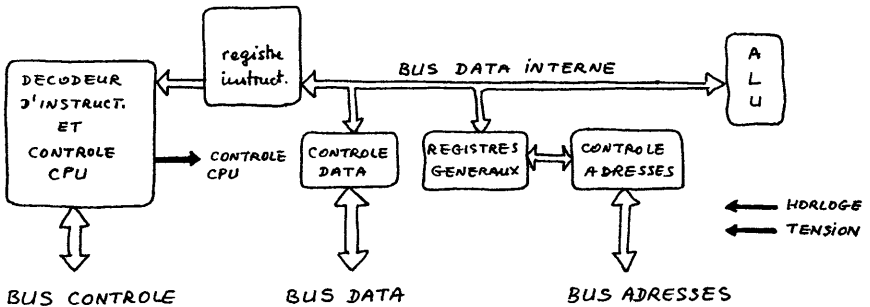
LE MATERIEL

Les différents types de circuits, outre les composants classiques : résistances, condensateurs, diodes et transistors, que nous trouverons sur la carte microprocesseur du TRS-80 peuvent être divisés ainsi : le microprocesseur, les mémoires, les opérateurs logiques, les circuits fonctionnels (bascules, diviseurs, décodeurs etc...).

Les technologies employées sont de type LS et CMOS qui sont caractérisées par leur faible consommation.

Le Microprocesseur

Comme vous le savez maintenant, c'est un Z 80. Il est schématiquement représenté comme ci-après :



C'est le plus gros circuit de la carte microprocesseur. Il possède 40 pattes qui lui permettent d'entretenir des rapports dictatoriaux avec son entourage.

Le Bus (ou canal de communication) d'adresses, permet de sélectionner les adresses mémoire nécessaires à l'exécution des ordres émanant de l'unité centrale. Il sera composé de 16 signaux. Les adresses peuvent également sélectionner un organe périphérique.

Le Bus de données, de 8 signaux, assure le transport de l'information entre les différents sous-ensembles reliés au microprocesseur. Ainsi, lorsqu'un octet est extrait de la mémoire par une instruction, il chemine sur ce Bus durant une fraction de temps, et est dirigé vers l'un des registres du Z 80.

Le Bus de commande (13 signaux), est utilisé pour mettre un peu d'ordre dans tout ce remue-ménage... Il indique par exemple le sens des informations (entrée ou sortie), le temps pendant lequel doit durer telle ou telle opération, l'organe de destination des données (mémoire ou périphérique), l'arrivée d'une interruption prioritaire etc... Maintenant que nous avons vu les pattes, voyons un peu à quoi elles sont rattachées.

Le coeur du microprocesseur est composé d'une unité logique et arithmétique (ALU) qui exécute les instructions demandées par le programme résidant en mémoire centrale. Pour ce faire, l'ALU communique avec un bloc de registres (mémoire auxiliaire très rapide) se trouvant dans le micro-processeur et servant d'intermédiaire entre ce dernier et le programme.

Ces registres, appelés registres généraux, sont au nombre de 22, et sont décrits ci-après.

BLOC PRIMAIRE

A (ACCUMULATEUR)	F (FLAGS-INDICAT.)
B	C
D	E
H	L
7	0 7 0

BLOC SECONDAIRE

A'	F'
B'	C'
D'	E'
H'	L'
7	0 7 0

BLOC SPECIAL

I (INTERRUPTION)	R (REFRESH)
IX (INDEX)	
IY (INDEX)	
SP (POINTEUR PILE)	
PC (COMPTEUR PROGRAMME)	
15	0

Enfin, un décodeur d'instructions a pour rôle de reconnaître le type d'instruction à exécuter, d'appeler éventuellement en mémoire centrale les informations qui y sont attachées et de sous-traiter son exécution à l'ALU.

Les mémoires

Deux types de mémoires sont utilisés dans le TRS-80 : mémoires ROM et RAM.

Mémoire R.O.M. (read only memory). C'est une mémoire dite "morte" qui, une fois écrite, conserve ses informations indéfiniment. Point question d'écrire à nouveau dans cette mémoire. Elle ne peut qu'être lue. Toute tentative d'écriture serait un coup d'épée dans l'eau, et les informations précédemment écrites resurgiraient toujours à la lecture.

Ce type de mémoire abrite le programme du moniteur et de l'interpréteur Basic. C'est lui qui est activé à la mise sous tension de l'appareil (départ à l'adresse ZERO).

Selon les versions logicielles Level I ou Level II, on trouve : soit deux boîtiers ROM de 2 K octets chacun, soit trois boîtiers ROM de 4 K octets chacun.

Signalons que la technologie actuelle permet des densités de l'ordre de 64 K bits par boîtier (8 K octets) pour ce type de mémoire.

On peut classer également dans les mémoires ROM, le générateur de caractères vidéo. C'est un circuit spécial qui contient le graphisme de tous les caractères apparaissant sur l'écran. Avez-vous remarqué que chaque caractère de l'écran vidéo est formé d'un ensemble de petits points ? C'est le générateur de caractères qui contient les informations permettant de tracer chaque caractère sur l'écran.

Mémoire R.A.M. (Random Access Memory). Elles sont divisées en deux types, tous deux utilisés dans le TRS-80 : RAM statiques et RAM dynamiques.

Les premières sont utilisées, dans le cas qui nous préoccupe, pour assurer la mémorisation des informations de l'écran vidéo (1024 caractères). Les *RAM statiques*, de par leur technologie, peuvent retenir une information tant que la tension d'alimentation reste établie, et tant qu'on ne vient pas réécrire une nouvelle information à la même place, bien sûr. La technologie actuelle permet des densités de 4 K bits par boîtier. Celles utilisées dans le TRS-80 ont une capacité de 1 K bits.

Les *RAM dynamiques*, elles, offrent une densité plus importante : 16 K bits (et bientôt 32 et 64... On n'arrête pas le progrès !). Mais il y a un petit détail : leur pouvoir de rétention de l'information est limité à quelques millièmes de secondes. C'est ce qui s'appelle "avoir la mémoire courte" ! Aussi, il faut "rafraîchir" ces mémoires (REFRESH) en lisant puis réécrivant les informations qu'elles contiennent, et ceci environ toutes les 2 ms (2 millièmes de secondes) par cellule mémoire.

Cette tâche est heureusement réalisée par un dispositif interne au Z 80 qui, chaque fois qu'il lit ou écrit en mémoire (ce qui ne manque pas de se produire), rafraîchit par la même occasion les mémoires dynamiques connectées à son Bus, et ceci sans la moindre perte de temps, puisque le rafraîchissement est effectué pendant les portions d'instructions n'utilisant pas la mémoire. Les RAM dynamiques constituent ce que nous appellerons la "mémoire centrale".

Selon la version qui équipe le TRS-80, 4 K octets en standard, ou 16 K en option, on aura affaire à deux types de boîtiers compatibles entre eux et interchangeables et montés fort heureusement sur supports enfichables (SOCKET) : 4 K bits en version 4 K octets (8 boîtiers), 16 K bits en version 16 K octets (8 boîtiers).

La mémoire RAM dynamique du TRS-80 peut, moyennant une "expansion interface" être amenée à un total de 48 K octets (16 K dans le clavier + 16 ou 32 K dans l'expansion interface). Ces mémoires (4 ou 16 K) possèdent 16 pattes de sortie. On remarquera que seulement 7 fils d'adresses sont utilisés, ce qui ne permettrait normalement d'adresser que 128 cellules. Or on en adresse réellement 16384 ! Quel est donc ce mystère ?

Pour réaliser ce petit tour de magie, on utilise le mode d'adressage dit "MULTIPLEXE", ce qui signifie que l'on envoie d'abord sur les 7 fils d'adresse, une adresse ligne, puis le temps d'après une adresse colonne. En fait, on envoie à la mémoire (mais en deux fois), l'équivalent de 14 fils d'adresse... Voilà le mystère qui tombe ! (bien entendu, pour les 4 K, 6 fils d'adresse seulement sont utilisés au lieu de 7).

Puisque nous en sommes au chapitre des mémoires, il est peut être bon d'en citer un autre type, non utilisé dans le TRS 80, mais qui pourra l'être par la suite, c'est la *mémoire EPROM*. Elle ressemble à la ROM, mais elle possède une petite "fenêtre" sur le dessus du boîtier, qui permet de l'effacer dans sa totalité lorsqu'elle est exposée aux rayons ultra-violets pendant 20 à 30 minutes. On peut ensuite y réécrire un nouveau programme à l'aide d'un appareil spécial.

Les capacités atteintes actuellement dans ce type de mémoire sont de 4 K octets (32 K bits). Comme pour la ROM, toute tentative d'écriture dans une EPROM par programme serait vaine. Elle est en principe utilisée pendant la phase de mise au point d'un logiciel qui risque d'évoluer ou d'être modifié avant la phase finale d'écriture en ROM.

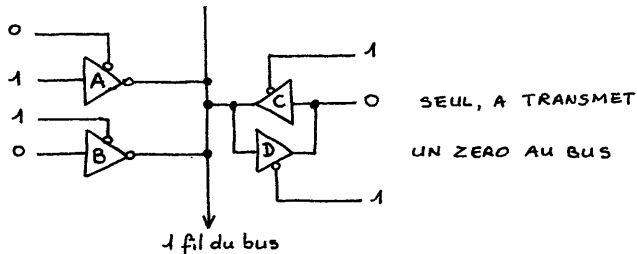
Les opérateurs logiques

Seront regroupés sous cette appellation les circuits électroniques réalisant des opérations booléennes élémentaires (et, ou, non-et, non-ou ; où, selon la terminologie anglo-saxonne : and, or, nand, nor). Les figures ci-contre montrent les principaux opérateurs et les différentes façons de les représenter.

Les amplificateurs et inverseurs trois états (TRI-STATE) sont représentés avec une porte de validation (GATE), en général commune à tous les circuits du même boîtier.

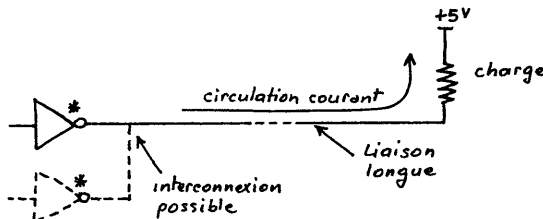
Lorsque cette porte est en niveau logique "1", la sortie du circuit est à haute impédance et elle n'a, de ce fait, aucune action sur les autres sorties qui lui sont reliées. Ces outils logiques sont en général employés dans le cas de raccordements aux Bus.

Par contre, lorsque la porte de validation est au niveau logique "0", l'amplificateur ou l'inverseur fonctionne normalement (voir figure ci-dessous).



Les mémoires sont elles-mêmes équipées d'une manière interne, d'entrées/sorties 3 états.

Il faut enfin signaler, parmi les circuits logiques un peu spéciaux, l'inverseur dit "à collecteur ouvert". Il ne se distingue de l'inverseur normal que par un point : la charge du transistor de sortie n'est pas intégrée dans le circuit, mais est fournie extérieurement.



$$\begin{array}{c} A \\ B \end{array} \rightarrow \text{OR} \rightarrow S = \begin{array}{c} A \\ B \end{array} \rightarrow \text{OR} \rightarrow S \quad S = A + B \text{ ou: } \overline{\overline{A \cdot B}}$$

CIRCUIT "OR" (OU)

$$\begin{array}{c} A \\ B \end{array} \rightarrow \text{NAND} \rightarrow S = \begin{array}{c} A \\ B \end{array} \rightarrow \text{NAND} \rightarrow S \quad S = \overline{A \cdot B} \text{ ou: } \overline{A} + \overline{B}$$

CIRCUIT "NAND" (NON-ET)

$$\begin{array}{c} A \\ B \end{array} \rightarrow \text{NOR} \rightarrow S = \begin{array}{c} A \\ B \end{array} \rightarrow \text{NAND} \rightarrow S \quad S = \overline{A + B} \text{ ou: } \overline{\overline{A \cdot B}}$$

CIRCUIT "NOR" (NON-OU)

$$\begin{array}{c} A \\ B \end{array} \rightarrow \text{AND} \rightarrow S = \begin{array}{c} A \\ B \end{array} \rightarrow \text{AND} \rightarrow S \quad S = A \cdot B \text{ ou: } \overline{\overline{A + B}}$$

CIRCUIT "AND" (ET)

CIRCUITS LOGIQUES ET LEURS REPRESENTATIONS

$$A \rightarrow \text{Buffer} \rightarrow S = A \rightarrow \text{Buffer} \rightarrow S \quad S = A$$

CIRCUIT AMPLIFICATEUR

$$A \rightarrow \text{Inverter} \rightarrow S = A \rightarrow \text{Inverter} \rightarrow S \quad S = \overline{A}$$

CIRCUIT INVERSEUR

$$\begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S = \begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S = \begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S \quad S = A \cdot \overline{G}$$

AMPLI TROIS-ETATS A VALIDATION

$$\begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S = \begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S = \begin{array}{c} A \\ G \end{array} \rightarrow \text{AND} \rightarrow S \quad S = \overline{A} \cdot \overline{G}$$

INVERSEUR TROIS-ETATS A VALIDATION

AMPLIS ET INVERSEURS LOGIQUES

Cela permet, par exemple, de relier entre elles plusieurs sorties "collecteur ouvert". Nous en verrons l'utilité lorsque nous aborderons le principe de fonctionnement du clavier.

Les circuits fonctionnels

Ce sont des circuits qui réalisent des fonctions de codage/décodage, de comptage, de division de fréquence, de sérialisation de données parallèles etc...

Ils sont composés, pour la plupart, de bascules et d'opérateurs logiques, et permettent un niveau d'intégration plus poussé.

LE LOGICIEL

Depuis le début de ce livre, votre regard est souvent tombé sur des mots tels que : binaire, langage-machine, langage évolué, octets, bits... etc. Selon notre conseil, vous êtes passé outre. Mais maintenant il est temps de s'expliquer un peu et, bien qu'il existe de nombreux ouvrages sur ce sujet, écrits par des spécialistes, nous allons essayer, non pas de définir, mais d'expliquer l'utilisation faite de ces mots.

D'abord, la notion de *langage*. Si vous essayez de converser avec un Chinois, vous ne tarderez pas à vous apercevoir qu'il serait vain de continuer le dialogue. De deux choses l'une : soit vous devez apprendre le Chinois, soit il doit apprendre le Français, à moins que vous décidiez (si toutefois vous arrivez à vous faire comprendre), de "couper la poire en deux" et d'apprendre une langue commune. C'est cette solution qui a été retenue avec l'ordinateur : il a été programmé dans sa langue maternelle (le langage machine), afin de comprendre un langage évolué (Basic) situé à mi-chemin entre celui de l'homme et de la machine (soyons honnête, plus près de l'homme que de la machine !). De votre côté, vous devrez faire l'effort nécessaire pour apprendre ce langage très simple.

Afin de donner à ce langage (le Basic), un caractère universel, le vocabulaire retenu a été l'Anglais (peut-être aussi parce que ceux qui l'ont fait les premiers parlaient cette langue). Mais, pour l'ordinateur, peu importe la langue. Un mot est, pour lui, une suite de caractères correspondant à une fonction précise. Son programme interprète (interpréteur Basic), décode, reconnaît cette suite de caractères et lui affecte les ordres correspondants mais dans le langage maternel du microprocesseur (le Binaire).

Prenons une analogie. Si l'on vous dit "écrivez le mot lune" (langage évolué), votre propre "programme", toujours en attente d'informations, captera par vos oreilles (organe d'entrée périphérique) et reconnaîtra ces mots (instructions). Il "interprète" alors cet ordre en le décomposant en une suite d'instructions plus fines telles que : tendre la main, prendre le crayon, s'approcher du papier, toucher le papier avec la pointe du crayon... etc (langage machine) qui, envoyées au cerveau (microprocesseur) sous forme codée, vont déclencher les stimuli nécessaires vers les muscles concernés (écriture, impression).

Avec l'ordinateur, il se passe à peu près la même chose mais, comme il n'a pas d'oreilles (tout au moins pas encore), il faudra frapper au clavier :

PRINT "LUNE"

suivi d'une touche spéciale indiquant la fin du message. Ça marche ! Le mot "LUNE" est apparu sur l'écran... Il a compris ! Le mot basic "PRINT" signifie en effet "IMPRIMER" en Anglais. Le vocabulaire Basic étant réduit, si vous frappez au clavier :

WRITE "LUNE"

(écrire "LUNE"), l'ordinateur ne comprendra pas car ce mot n'existe pas en langage Basic, ce qui entraînera de la part de l'interpréteur une injure du style "ERREUR DE SYNTAXE"...

Passons maintenant au langage dit "ASSEMBLEUR". C'est le langage propre à chaque ordinateur et généralement différent d'un ordinateur à l'autre, mais exprimé sous forme symbolique. Pour revenir à notre analogie, c'est un peu comme si on vous avait dit, au lieu de "écrivez lune" : "tendez la main, prenez le crayon, approchez vous du papier, ... etc"

Chaque microprocesseur a "son" assembleur, Z 80, 8080, 6800, 6502...

A chaque instruction symbolique du langage d'assemblage correspondra une, et une seule, instruction "machine" élémentaire. Cette conversion sera réalisée par une sorte d'interpréteur également, nommé "le programme assembleur".

L'ordinateur ne comprenant réellement que le Binaire, c'est à dire une succession de 1 et de 0, il est plus pratique d'écrire :

LD A,5

ce qui signifie : chargement de la valeur 5 dans le registre 'A' (c'est la forme symbolique), plutôt que :

0011111000000101.

Une forme plus contractée serait d'écrire :

3E05

ce qui a la même signification, mais là le système de numérotation n'est plus binaire (base 2) mais hexadécimal (base 16).

Résumons-nous :

LD A,5	0011111000000101	3E05
=	=	=
SYMBOLIQUE	BINAIRE	HEXADÉCIMAL

Voici 3 façons de représenter la même chose. Laquelle des trois préférez-vous ?

Lorsque l'on sait que "LD" signifie "LOAD", c'est-à-dire "chargement" en Anglais, la forme "LD A,5" parle toute seule : Charger A avec la valeur 5.

BINAIRE, HEXADECIMAL : qu'est-ce que c'est ? Ce sont des *systèmes de numérations*.

Pourquoi, lorsque nous additionnons 1 à 9, devons-nous mettre un 0 au chiffre des unités et un 1 au chiffre des dizaines ? Parce que nous n'avons que 10 symboles (0 à 9) pour compter. Nous comptons en décimal, autrement dit en base 10. Au delà de 9, il n'y a plus d'autres symboles, il faut donc repartir à 0 et ajouter un chiffre à gauche.

En *BINAIRE* (base 2), il n'y a que deux symboles : 0 et 1, parce que c'est plus facile pour un ordinateur (un trou/pas de trou ; un signal/pas de signal ; une magnétisation/pas de magnétisation). Au delà de 1, il n'y a pas de 2. Il faut donc repartir à 0 et ajouter le 1 à gauche, ce qui fait :

```

  1
+ 1
---
```

10

continuons :

```

  10
+ 1
---
```

11

pas de problème...

```

  11
+ 1
---
```

100

Logique, non ?

4 font donc 100 en Binaire.

De même, 15 font 1111.

Mais là où cela se complique, c'est lorsqu'il faut dire que :

40697 font 1001111011111001.

C'est assez encombrant... Aussi les informaticiens ont-ils décidé d'exprimer la même chose, mais sous une forme plus simple, en utilisant la base 16, c'est-à-dire exprimée avec 16 symboles.

En *HEXADECIMAL*, cette fois 9 + 1 ne font pas 10, mais 'A' (le dixième symbole de la base). L'astuce a donc été de remplacer les symboles manquants au delà de 9 par les premières lettres de l'alphabet 'A' à 'F'. 9 + 1 font 10 en base 16, et 40697 en base 10 font 9EF9 en base 16, ce qui se représente 9EF9H, le 'H' signifiant 'hexadécimal'.

Pour passer du binaire à l'hexadécimal, il suffit d'apprendre par coeur les 16 premiers nombres et leurs correspondances hexadécimales et décimales :

BINAIRE	HEXA	DECIMAL
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

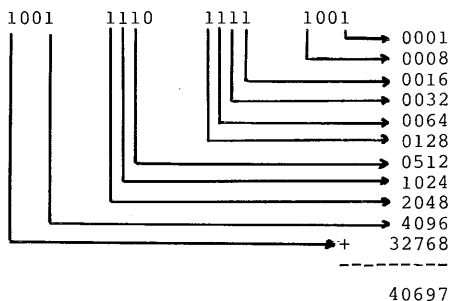
Le nombre 1001111011111001 est ensuite divisé en tranches de 4 chiffres en partant de la droite :

1001	1110	1111	1001
9	E	F	9

Voilà ! Ce n'est pas plus difficile !

Bien entendu, l'ordinateur ne comprend pas plus l'hexadécimal que l'assembleur; mais le programme qui permet de passer de l'hexadécimal au binaire est très simple.

Pour passer du binaire au décimal, il suffit de remonter les puissances de 2 en partant de la droite et d'additionner toutes celles dont le chiffre correspondant est à 1. Exemple :



Et maintenant : ce qu'on appelle un *bit*, un *Octet* ?

Un bit, c'est l'élément binaire prenant la valeur 1 ou 0.

Dans les mémoires de l'ordinateur, les nombres sont stockés sous forme de bits et un ensemble de 8 bits est appelé un octet. Les instructions Basic, assembleur sont toutes codées en mémoire sous forme de bits et d'Octets. Le nombre ci-dessus sera représenté en mémoire '100111101111001' (sur 16 cases mémoires).

Dans un programme, autant que dans le fonctionnement intime de l'ordinateur, on est souvent conduit à exécuter des opérations spéciales sur les valeurs binaires. Les *opérations logiques* de base sont au nombre de 3 : la *complémentation* (NOT), le '*ET*' logique (AND) et le '*OU*' logique (OR).

La complémentation c'est prendre le complément ou inverse de la valeur du bit correspondant. Exemple : le complément de la valeur binaire 1011 et 0100.

Les nombres manipulés par le Basic étant ordinairement représentés sur 16 bits, le dernier bit de gauche (appelé poids fort, plus forte puissance de 2), représente le signe du nombre (positif si 0, négatif si 1). Dans ce cas, le complément de 1011 sera 111111111110100, c'est-à-dire -12 en décimal, puisque si l'on ajoute 12 à cette valeur, on repasse à zéro :

1111	1111	1111	0100
+ 0000	0000	0000	1100

0000	0000	0000	0000

(le 1 encadré ne compte pas, on dit qu'il a débordé).

En Basic, si vous frappez : PRINT NOT 11 ; l'ordinateur vous répondra : -12.

Mais attention : les nombres manipulés par les opérateurs logiques ne peuvent dépasser la valeur 32767 (15 bits en binaire).

Le '*ET*' logique : ici, le résultat logique de 2 bits égaux à 1 sera toujours 1. Dans les autres cas, il prend la valeur 0.

Exemple :

	01011	(=11)
'ET'	00110	(= 6)
	00010	(= 2)

En basic, PRINT 11 AND 6, donnera 2.

Le 'OU' logique : dans ce cas, le 1, s'il est présent l'emporte toujours. Exemple :

	0	1	0	1	1	(=11)
'OU'	0	0	1	1	0	(= 6)
	0	1	1	1	1	(=15)

En basic, PRINT 11 OR 6, donnera 15.

En Basic, comme dans tout langage de programmation évolué, les *nombre*s sont représentés dans le système décimal. Selon la notation anglo-saxonne, la virgule est remplacée par le point décimal. Ainsi : 2,25 s'écrit : 2.25

Un nombre est dit *entier* lorsqu'il n'a pas de point. Exemples : 3, 25, -1246

Tout nombre ayant un point est dit *réel*. Exemples : 1.25, .5, 1., -2.4

En Basic II, un nombre peut être défini avec l'un des deux types de *précision* : la *simple* ou la *double* (elles sont respectivement notées ! et # en Basic).

En simple précision, 6 chiffres au maximum peuvent être utilisés, alors qu'en double précision, 16 sont permis.

Une façon pratique de représenter de très petits ou de très grands nombres est de les écrire en *notation scientifique* : 2.25E9 signifie qu'il faut déplacer le point décimal de 9 chiffres vers la droite ou la gauche, selon le signe de l'exposant, pour obtenir la valeur réelle du nombre.

2.25E9 = 2 250 000 000

2.25E-9 = 0.000 000 002 250

En double précision, on utilise le 'D' en place du 'E'.

CHAPITRE 4

LE BASIC II

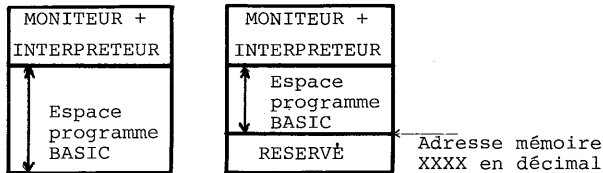
Introduction pratique

Il ne sera pas question ici du Basic niveau 1, cette version étant un sous-ensemble de celle que nous allons examiner ci-après. Nous nous contenterons d'en rappeler les commandes en annexe.

Pour pratiquer le Basic "sur le tas", le mieux est d'abord de mettre la machine sous tension et de pianoter un peu. N'ayez aucun scrupule : étant très patient de nature, l'ordinateur ne se vexera pas s'il ne comprend pas toujours ce que vous essayez de lui dire. Il se bornera, dans ce cas, à répéter inlassablement "SYNTAX ERROR" ou quelque chose du même style.

Dès la mise sous tension, ce qui semble être une question apparaît sur l'écran : MEMORY SIZE?

Kekcekaç MEMORY SIZE? C'est lui qui pose des questions maintenant ? Ne vous paniquez pas. Il veut tout simplement vous demander la taille maximum de l'espace mémoire que vous voulez réserver pour les programmes Basic, ceci dans le cas où vous voudriez charger en mémoire un programme binaire devant coexister avec un programme Basic.



MEMORY SIZE? <ENTER> MEMORY SIZE? XXXX <ENTER>

Dans le cas présent, mieux vaut répondre en appuyant sur la touche 'Enter', sachant que ces questions-là seront examinées en détails dans le volume II plus spécialement consacré au langage machine.

Après un court instant, variable selon la taille mémoire dont vous disposez, un message sécurisant apparaît :

➤READY

Ce qui signifie quelque chose comme : "Je suis prêt, maître, j'attends vos ordres."

A noter que, sous Basic 2, le signe ">" en début de ligne a toujours ce sens lorsqu'il est affiché par le moniteur. Il faut savoir également que tout ordre transmis à l'ordinateur sera toujours terminé par la touche 'Enter' qui correspond à 'Fin de message'. Aussi, dans ce qui suit, toute commande ou ordre sera terminé par cette touche, même si cela ne figure pas explicitement dans les exemples.

Ordonnons donc, puisque nous pouvons donner des ordres :

PRINT "LUNE"

Le message "LUNE" est imprimé sur l'écran (voir chapitre précédent).

Continuons :

PRINT LUNE

Le chiffre 0 est imprimé. Pourquoi ?

Dans le premier cas, le mot LUNE était entouré de guillemets, ce qui indiquait une chaîne de caractères (un texte) que l'ordinateur ne devait pas toucher (interpréter).

Dans le second cas, il n'en est pas de même. L'interpréteur essaye donc de comprendre un peu, et recherche dans tous ses " tiroirs de rangement " celui étiqueté "LUNE". Il le trouve et en imprime le contenu. Mais, comme tous les tiroirs ont été mis à zéro au départ, il imprime donc 0.

Essayons maintenant :

```
LUNE = 1234
PRINT LUNE
```

Ca marche ! La première commande a chargé la valeur 1234 dans le tiroir étiqueté LUNE. La seconde commande a imprimé le contenu du tiroir LUNE.

Essayons encore :

```
PRINT LUNE    (affichage de : 1234)
PRINT LUGE    (affichage de : 1234)
```

Tiens ? Il y a des tiroirs qui communiquent entre eux ?

Non ! En fait, l'explication est toute simple : le nom de chaque tiroir ne possède que deux caractères au maximum, le reste des caractères au delà de deux est tout bonnement ignoré.

Pour l'interpréteur, LUNE et LUGE désignent le même tiroir. Il faut le savoir, c'est tout !

Si vous avez compris cette notion de tiroir, et bien vous avez compris la notion de "variable", qui est le mot plus "scientifique" consacré à cette appellation. Une variable est donc un tiroir qui peut contenir des choses variables...

Poursuivons notre idée : il y a plusieurs sortes de tiroirs :

- des tiroirs ne pouvant recevoir que des nombres (variables numériques), eux-mêmes divisés en quatre catégories : les tiroirs à nombres entiers, les tiroirs simples à 6 chiffres (simple précision), les doubles, à 16 chiffres, (double précision) et les tiroirs à casiers (variable indicée).
- des tiroirs pouvant recevoir des caractères alphanumériques (chiffres, lettres, caractères spéciaux). On les appelle des variables de chaîne.

Pratiquons un peu avec une "variation sur le nombre PI". Vous connaissez tous le nombre PI ; l'ordinateur, lui, ne le connaît pas. Pour le lui apprendre, frappons au clavier :

```
PI = 3.1416    (mettre 3.1416 dans le tiroir PI)
```

Ca y est ! Il le connaît maintenant. La preuve :

```
PRINT PI      déclenche le message : 3.1416
```

Mais PI sur 2, ça fait combien ?

```
PRINT PI/2    (le signe / est employé pour 'diviser')
1.5708        ... dit-il. Correct
```

Voulons-nous conserver cette valeur dans un autre tiroir (que nous appellerons P2, par exemple ?) Rien de plus simple :

```
P2 = PI/2
```

Mais jusqu'ici, nous n'avons pas encore réellement "écrit" un programme. Nous ne faisons qu'utiliser l'ordinateur en mode 'calculateur' (par opposition au mode 'programme'). Un programme : ce sera une suite d'instructions identiques à celles que nous avons exécutées jusqu'à présent, mais précédées d'un numéro de ligne. Ces numéros seront quelconques, la seule règle étant de respecter un ordre chronologique croissant dépendant de l'ordre d'exécution des instructions.

Frappons donc notre premier programme au clavier :

```
10 PI = 3.1416
20 PRINT PI
```

remarquez qu'ici il ne se passe rien, l'ordinateur attend l'ordre pour exécuter le programme avant d'imprimer. Continuons :

```
30 P2 = PI/2
40 PRINT P2
```

Voilà ! Maintenant, exécutons notre programme en frappant l'ordre adéquat (lancement de programme) :

```
RUN
```

Si vous n'avez pas fait d'erreur (sinon vous devrez refrapper la ou les lignes erronées), l'écran doit afficher :

```
3.1416
1.5708
```

suivi du message :

```
READY
>
```

Mais cela n'est pas très parlant. Frappez :

```
20 PRINT "PI=" ; PI
40 PRINT "PI SUR 2=" ; P2
```

Les lignes 20 et 40 sont automatiquement remplacées.

```
RUN
```

Vous devez avoir cette fois :

```
PI= 3.1416
PI SUR 2= 1.5708
```

Remarquez les signes ";" des lignes 20 et 40. Ils servent à séparer les différents champs d'impression. Essayez d'en mettre un à la fin de la ligne 20. Vous aurez :

PI= 3.1416 PI SUR 2= 1.5708

Remplacez-le maintenant par une virgule :

PI=3.1416 PI SUR 2= 1.5708

C'est ce que l'on appelle une "tabulation", très utile pour la mise en page des tableaux. Le signe "," est aussi un séparateur.

Voilà qui vous donne un avant-goût de ce que peut être la programmation en Basic sur le TRS-80. Une chose reste à faire maintenant : apprendre à connaître les différents mots du vocabulaire Basic. Les pages suivantes vous y aideront, mais ne restez pas dans la théorie : Pratiquez... ! Pratiquez... ! C'est le secret pour l'apprentissage d'un langage, qu'il soit humain ou non.

Bien sûr, vous ferez probablement des erreurs : c'est parfait ! c'est le métier qui rentre ! Cherchez à les comprendre et à les corriger.

Avant de voir les différents mots utilisés dans un programme Basic, nous allons passer en revue les commandes du moniteur et les fonctions qu'elles réalisent.

COMMANDES MONITEUR

RUN

Vous l'avez déjà utilisée. C'est elle qui permet de lancer l'exécution d'un programme Basic chargé en mémoire. Deux formes sont possibles :

RUN

RUN numéro de ligne

Dans le premier cas, le programme est lancé à la première ligne, alors que dans le second il est lancé à partir d'un numéro donné exemple :

RUN 20

C'est l'occasion de remarquer que chaque commande RUN est toujours précédée d'une remise à zéro des " tiroirs ". En effet, la ligne 10 du programme n'ayant pas été exécutée, les valeurs de PI et de P2 seront trouvées égales à zéro.

LIST

Essayez de frapper cette commande, et vous en comprendrez immédiatement l'utilité (si le programme précédent est toujours en mémoire). Cette commande vous donne en effet la liste des lignes formant le programme présent en mémoire.

Si le programme est long, la liste peut être momentanément suspendue par l'appui sur les touches SHIFT et @ . La prochaine touche enfoncée fait progresser la liste, etc...

Naturellement la touche BREAK peut être employée, mais elle interrompt la liste définitivement.

Cette commande peut prendre plusieurs formes :

LIST (liste à partir de la première ligne)

LIST n° ligne (liste de la ligne spécifiée)

LIST n1 - n2 (liste de la ligne n1 à la ligne n2)

A noter que n1 ou n2 peut être omis. Essayez pour voir...

NEW

Frappez cette commande au clavier. Exécutez ensuite la commande LIST. L'ancien programme a tout simplement disparu. Dommage ! après tout ce temps passé à le rentrer ! Vous avez compris maintenant que cette commande est à utiliser avec préméditation.

AUTO

Voici une facilité qui vous permettra de ré-entrer votre programme. Faites : AUTO ENTER. Le numéro de ligne 10 est affiché automatiquement. Frappez ensuite la première ligne du programme :

PI = 3.1416 <ENTER>

et ainsi de suite jusqu'à la dernière ligne. Pour arrêter, appuyez sur la touche BREAK. Faites : LIST.

Votre programme doit être là de nouveau.

Petit gadget : si en cours de commande AUTO, un caractère "*" est affiché à droite du numéro de ligne, cela signifie que cette ligne est déjà occupée. A vous de décider si vous voulez la remplacer ou non.

La commande AUTO peut prendre les trois formes :

AUTO (départ ligne 10 et de 10 en 10)

AUTO n (départ ligne n et de 10 en 10)

AUTO n,m (départ ligne n et de m en m).

CONT

Cette commande signifie "continuer". Elle est à utiliser dans le cas où le programme a été stoppé dans son déroulement par la touche BREAK ou l'ordre STOP (voir plus loin). Cette commande permet donc de relancer le programme au point interrompu.

DELETE

Signifie : supprimer, effacer. Cette commande permet de supprimer une ou plusieurs lignes dans un programme Basic. Si vous voulez toutefois supprimer une seule ligne, ou des lignes non contiguës, il est plus pratique de frapper le numéro de la ligne à effacer suivi de ENTER (une ligne vide est automatiquement rejetée du programme). Dans l'autre cas, vous frapperez :

DELETE n1-n2

pour supprimer la portion de programme comprise entre les lignes n1 et n2 incluses. Si n1 est omis, la portion supprimée commencera à la première ligne.

CSAVE

Voilà la commande qui vous permettra de sauver le contenu d'un programme Basic sur cassette. Avant de lancer cette commande, il faudra toutefois veiller que le magnétophone soit bien en

position écriture (boutons Record et Play enfoncés) et que la cassette soit positionnée après l'amorce non magnétisée de la bande. La forme de cette commande est la suivante :

CSAVE "n"

"n" étant un caractère alphanumérique indiquant le nom à donner au programme à sauver. Pendant toute la durée de l'écriture cassette, le clavier est inhibé : aucune frappe n'est prise en compte. Attendez que le message READY soit affiché sur l'écran. Essayez maintenant de sauver le petit programme sur le nombre PI.

CLOAD?

Cette commande permet de vérifier que votre programme Basic a bien été correctement écrit sur la cassette. Une lecture de la cassette est déclenchée, et chaque Octet lu est comparé avec son équivalent en mémoire. En cas d'erreurs de comparaison, un message apparaît :

BAD (c'est-à-dire : MAUVAIS)

Cette commande est de la forme :

CLOAD? "n"

ou "n" est le nom du programme à comparer (voir CSAVE) dans le cas où celui-ci n'est pas en première position sur la cassette. Sinon "n" peut être omis.

CLOAD

Cette commande permet de lire un programme sur cassette et de le transférer en mémoire. Dès son lancement, une commande NEW est automatiquement exécutée, afin de libérer la zone devant recevoir le programme. Deux formes possibles :

CLOAD (Chargement du premier programme lu sur la cassette)

CLOAD "n" (recherche et chargement du programme n)

Dans le second cas, tout programme lu sur la cassette et ne correspondant pas au nom "n" est sauté (le nom est affiché sur l'écran pendant ce temps), et ainsi de suite jusqu'à trouver le bon nom. Pendant la lecture, un astérisque situé en haut à droite de l'écran clignote selon un temps dépendant de la longueur des lignes Basic.

SYSTEM

Cette commande est, comme la précédente, une commande de lecture cassette, mais réservée à la lecture des programmes Binaires.

Certains programmes, comme l'éditeur/assembleur EDTASM et TBUG sont chargés ainsi. Après la frappe de cette commande, le message "*?" apparaît sur l'écran. Le nom du programme à charger doit alors être introduit (6 caractères sont autorisés), suivi de "Enter". La lecture cassette commence alors, signalée par un "*" fixe et un clignotant (haut droit de l'écran).

En cas d'erreurs de lecture, le caractère 'C' prend la place de l'un des astérisques. Il faut alors reprendre la lecture à son début.

En cas de réussite, et lorsque l'ordinateur affiche de nouveau "*?", répondre :

- /ENTER pour lancer l'exécution au début du programme ;
- /Adr Enter, adr étant la valeur décimale d'une adresse mémoire de lancement du programme.

Nous n'insisterons pas davantage sur cette commande, sauf pour dire qu'elle peut être utilisée aussi pour revenir au message initial MEMORY SIZE sans être obligé de couper l'alimentation.

SYSTEM Enter

*? /0 Enter

LA COMMANDE D'ÉDITION : EDIT

Cette commande permet essentiellement de modifier une ligne Basic sans être obligé de la refrapper entièrement. La forme d'appel est la suivante :

EDIT numéro de ligne Basic

Dès la frappe de cette commande (suivie de la touche Enter), et à condition, bien entendu, que la ligne spécifiée soit présente, l'écran affiche le numéro de ligne. Vous êtes maintenant sous le contrôle de la commande EDIT.

Pour continuer le dialogue, vous devez utiliser un certain nombre de sous-commandes, selon la fonction désirée.

Voici la liste de ces sous-commandes :

n barre d'espace : avance curseur de n caractères.

← ou n← : Recul d'un ou de n caractères par rapport au curseur.

L : Liste la ligne éditée.

I : Commande d'insertion. Tout caractère frappé derrière cette sous-commande est inséré à l'endroit pointé par le curseur.

D ou nD : Suppression d'un ou de n caractères après le curseur. Le champ supprimé est alors délimité par 2 caractères "!".

C ou nC : Remplacement (change) d'un ou de n caractères situés après le curseur.

Sc : Recherche du caractère "c" dans la ligne éditée, avec arrêt à gauche de ce caractère. Cette sous-commande peut prendre aussi la forme qui suit :

nSc : Recherche du n ième caractère "c". Nota : ne fonctionne pas si le caractère à chercher est le premier de la ligne.

Kc (ou nKc) : Suppression de tous les caractères situés avant le caractère "c" (ou le n ième caractère "c").

X : Insertion en fin de ligne. Le curseur va en fin de ligne et l'on passe en mode insertion (identique à I).

H : Suppression des caractères suivant le curseur, et passage en mode insertion.

E : On quitte le mode d'édition en apportant les corrections précédentes à la ligne éditée.

Q : On quitte le mode d'édition, mais sans modifier la ligne (les corrections ne sont pas apportées).

A : Effacement des corrections précédentes éventuelles et retour au début de la ligne éditée. Pour que cette sous-commande soit effective, on ne doit pas être engagé dans une autre sous-commande. Sinon, frapper ce qui suit :

Shift ↑ : Cette sous-commande permet de sortir d'une autre sous-commande (I par exemple), mais sans quitter le mode EDIT.

Enter : Toutes les modifications sont enregistrées. On quitte le mode EDIT.

LES SYMBOLES DE BASE

Comme tous les langages évolués, le Basic est construit à partir d'un ensemble de symboles de base, dit "alphabet". Cet ensemble est simple, et varie très peu d'un constructeur à un autre. Il est composé :

- de chiffres	0, 1, ..., 9
- de lettres majuscules	A, B, ..., Z
- d'opérateurs arithmétiques	
. addition	+
. soustraction	-
. multiplication	*
. division	/
. exponentiation	↑
- des parenthèses gauche et droite	()
- d'opérateurs relationnels	
. égal	=
. inférieur à	<
. supérieur à	>
. inférieur ou égal à	< = ou =<
. supérieur ou égal à	> = ou =>
. différent de	<> ou <>
- d'opérateurs logiques	
. opérateur "ET"	AND
. opérateur "OU"	OR
. opérateur "NON"	NOT
- de caractères spéciaux	
. guillemet	"
. dollar	\$
. dièse	#
. point d'exclamation	!
. pourcent	%
. et commercial ou perluète	&
. apostrophe	'
. point d'interrogation	?
. point	.
. deux points	:
. virgule	,
. point virgule	;
. a "escargot" ou a commercial	@
. blanc ou espace	

Tout le langage Basic est construit à partir de cet alphabet et comporte essentiellement des notions :

- de constantes,
- de variables,
- de fonctions,
- de mots réservés pour définir les instructions.

LES CONSTANTES ET LES VARIABLES

Il existe deux types de constantes et de variables :

Les constantes "numériques" qui sont représentées en virgule flottante et les variables "numériques"

Les constantes "chaîne" et les variables "chaîne" ou "alphanumérique" (String en américain).

Constante numérique :

Une constante numérique est représentée sous l'une des trois formes suivantes :

- entière : -3,245
- décimale : 3.141592654 0.654
- décimale avec exposant : .3145926E+01

Lorsque la constante est positive, le signe + peut être omis.

Par contre, pour représenter un nombre négatif, il est impératif de faire précéder la constante du signe -.

Constante chaîne :

Une constante chaîne est une suite de caractères quelconques bornée par les caractères spéciaux "guillemets".

"TRS-80" "BASIC" , "1234567890"

sont des constantes "chaîne". En Basic II, une constante chaîne peut avoir une longueur de 0 à 255 caractères au maximum (" est une constante de chaîne nulle).

Variable :

Une variable est désignée par un nom (ou identificateur de variable) formé par une suite de caractères alphanumériques, le premier devant être obligatoirement alphabétique (A-Z).

Exemple : A, CX, F5 TOTAL ; pourraient être des noms de variables.

En Basic II, seuls les deux premiers caractères sont significatifs. Une petite remarque au passage : éviter absolument les identificateurs de variables renfermant un mot réservé. (La liste de tous les mots réservés est donnée en annexe).

Pour définir la nature d'une variable, on écrit à droite du nom de la variable un caractère spécial : ce caractère tient lieu d'attribut.

Le Basic du TRS-80 reconnaît quatre sortes d'attributs :

```
% pour une variable entière           : A% AB% C%
! pour une variable simple précision : A! AB! C!
# pour une variable double précision : A# AB# C#
$ pour une variable de chaîne         : A$ AB$ C$
```

Si un nom de variable n'est pas suivi d'un attribut, elle est considérée, par défaut, comme étant du type simple précision.

Exemple : D4 et D4!

Déclaration de variables

Déclarer une variable, c'est définir une fois pour toutes son type. Il existe à cet effet, dans le Basic, quatre mots réservés pour définir les attributs d'une variable ou d'une liste de variables.

Cette possibilité, offerte par le Basic du TRS-80, permet d'alléger considérablement l'écriture des variables et d'optimiser l'occupation mémoire du programme.

```
DEFINT : pour déclarer une ou plusieurs variables entières.
DEFSNG : pour déclarer une ou plusieurs variables simple précision.
DEFDBL : pour déclarer une ou plusieurs variables double précision.
DEFSTR : pour déclarer une ou plusieurs variables de chaîne.
```

Exemple :

```
DEFINT A
```

Toute variable commençant par la lettre A est considérée comme entière : A, A6, AJ, ATCK... sont des variables entières.

```
DEFINT A-I
```

défini comme entières toutes les variables commençant par une des lettres A, B, C, D, E, F, G, H, I. A1, C3, I2, BB... sont des variables entières.

```
DEFINT C,K,M
```

Seules les variables commençant par une des lettres C, K, M sont considérées entières. CT, K2, MR, KLE... sont des variables entières.

```
DEFINT A,G,I-Z
```

déclare comme entière toute variable commençant par une des lettres A, G, et de I à Z comprise.

La même règle est appliquée à DEFSNG, DEFDBL et DEFSTR.

Il faut cependant noter que l'on ne peut rencontrer la même lettre dans des déclarations de types différents. Ainsi

```
DEFINT A-D
```

```
DEFSTR C
```

La deuxième ligne n'est pas correcte, car la variable C a déjà été déclarée entière ; à moins que ce ne soit intentionnel.

Les variables indicées

On désigne fréquemment un ensemble de quantités de même nature par un nom unique, auquel on ajoute des indices, comme en mathématiques par exemple, les n composants d'un vecteur V dans un espace à n dimensions seront dénommés : $V(1)$, $V(2) \dots V(n)$. De même, les $n \times p$ éléments d'une matrice M seront dénommés : $M(1,1)$, $M(1,2), \dots, M(1,p)$, $M(2,1)$, $M(2,2), \dots M(n,p)$.

L'ensemble de tous ces éléments s'appelle un tableau. En Basic, une variable indicée, donc un élément de tableau, se reconnaît par la présence, à droite de l'identificateur de la variable, d'une expression entre parenthèses, qui caractérise l'indice.

Pour faire référence à une variable indicée, par exemple $B(6)$, dans une expression arithmétique, nous écrirons :

```
...B(6)...
```

L'indice peut être lui-même une variable :

```
...B(I)...
```

L'indice peut, lui-même, être une variable indicée :

```
...B(I(J))
```

Ici, l'indice de B est représenté par $I(J)$.

L'indice d'une variable est toujours une constante entière positive, supérieure ou égale à zéro.

Si l'indice n'est pas une valeur entière, la valeur qui est considérée représente la partie entière de l'indice (troncature et non pas arrondi).

Exemple :

```
100 I=15.9
```

```
120 A(I)=5.78
```

```
140 X=A(I)
```

La valeur prise pour $A(I)$ sera $A(15)$, et le résultat dans X sera 5.78.

Tableau à N dimensions.

Considérons le schéma suivant :

	TABLEAU -A-		TABLEAU -B-
	COLONNE 0		COLONNE 0 COLONNE 1
LIGNE 0	:-----: : A(0) : :-----:		:-----: :-----: : B(0,0) : B(0,1) : :-----: :-----:
1	: A(1) : :-----:		: B(1,0) : B(1,1) : :-----: :-----:
2	: A(2) : :-----:		: B(2,0) : B(2,1) : :-----: :-----:
3	: A(3) : :-----:		: B(3,0) : B(3,1) : :-----: :-----:
LIGNE 4	: A(4) : :-----:		: B(4,0) : B(4,1) : :-----: :-----:

Le tableau -A- représente un ensemble de 5 éléments A(0) à A(4), rangés sur une seule colonne. On dit que c'est un tableau à une dimension.

Le tableau -B- représente un ensemble de 10 éléments, rangés sur 5 lignes et sur 2 colonnes B(0,0) ... B(4,1). On dit que c'est une matrice de 5 lignes par 2 colonnes.

Il n'y a, en principe, pas de limites au nombre de lignes et de colonnes d'un tableau. Ces caractéristiques sont appelées communément : la dimension d'un tableau.

Dans un programme Basic, lorsque l'on veut utiliser un tableau, il est impératif de définir sa dimension.

L'instruction DIM permet de déclarer la dimension d'un tableau, quelle que soit sa nature (! # % OU \$)

La forme d'écriture de cette instruction est la suivante :

DIM nom (DIM1,DIM2,...,DIMK)

nom représente le nom donné au tableau.

DIM1,...,DIMK les dimensions du tableau.

Pour définir l'exemple précédent, on écrirait :

10 DIM A(5)

20 DIM B(5,2)

Une même instruction DIM peut déclarer plusieurs tableaux.

Exemple :

10 DIM A(5),B(5,2),C(3,2,5)

Les variables de "chaîne" peuvent également être indicées.
Si le tableau -A- était un tableau de chaînes, on le définirait ainsi :

```
10 DIM A$(5)
100 A$(4)="TRS-80"
140 X$=A$(4)
```

La variable X\$ prendra pour valeur la chaîne "TRS-80".

Remarque importante et intéressante sur les tableaux

Un tableau peut avoir le même nom qu'une variable non indicée, ce sont deux notions différentes. Si l'on écrit :

```
10 DIM A(5)
20 A(3) = 10 (10 dans le 3ème élément du tableau A)
30 A = 3
```

la variable A contiendra 3, et n'a rien de commun avec le tableau A.

L'INSTRUCTION D'AFFECTION = et LET

Cette instruction permet d'affecter la valeur d'une expression à une variable. Le symbole d'affectation que nous avons d'ailleurs déjà rencontré est le caractère égal (=).

L'affectation consiste à donner à la variable citée à gauche du = la valeur de l'expression qui se trouve à sa droite.

Quand vous écrivez en mathématiques : $C=10$, vous affectez la valeur 10 à la variable C ; de même, si vous écrivez $X=A+(B*B)$, vous affectez la valeur de l'expression $A+(B*B)$ à la variable X. Et bien, en Basic, c'est la même chose.

Vous écrivez $A=10$, $B4\# = 2.4567$, $I\% = 8$, $X=A+(B*B)$ et aussi $C\$="TRS-80"$; dans ce dernier cas, la valeur affectée à la variable de chaîne C\$ est représentée par la chaîne de caractères "TRS-80". Les caractères guillemets qui encadrent TRS-80 ne font pas partie de la chaîne. Ils ont pour but de la délimiter.

L'instruction LET a le même effet, mais l'écriture est un peu différente.

On écrira

```
LET A=10, LET A$="TRS-80"
```

tout comme

```
A=10 ou A$="TRS-80"
```

Ce qui signifie, en clair : Faire A=10 et Faire A\$="TRS-80". Cette forme d'écriture est moins utilisée. Elle permet d'assurer une compatibilité avec d'autres Basic qui n'admettent pas l'affectation précédente. Aussi, pour des raisons évidentes de simplification, nous utiliserons la première notation.

Nous venons de voir l'affectation d'une constante à une variable, on peut aussi affecter une variable à une autre variable.

Exemple :

```
A=10
B2=A          ; B2 aura pour valeur 10
de même C$="TRS-80"
B2$=C$        ; B2$ aura pour valeur TRS-80.
```

Par contre, B2\$=A est une forme d'écriture incorrecte, car B2\$ est une variable de chaîne et "A" une variable simple précision ; sauf si "A" a été déclaré par DEFSTR.

```
DEFSTR A-D
X$="TRS-80"
A=X$          ; est correct
D=A           ; est également correct
en outre A#=12.45
C=A#          ; est correct, car la variable double
précision A# est convertie en variable simple précision.
```

Les conversions de ce type sont possibles avec le Basic du TRS-80 mais ce n'est pas le cas pour la plupart des Basics !

Un petit conseil, prenez garde aux troncatures et aux arrondis, quand une variable est convertie dans une variable de précision inférieure.

Si, par exemple :

```
A#=1.23456789
B=A#          ; B aura pour valeur 1.23457
C%=A#         ; C% aura pour valeur 1.
```

EVALUATION DES EXPRESSIONS ARITHMETIQUES

Comme en mathématiques, les parenthèses jouent en programmation un rôle très important. Les expressions arithmétiques peuvent aller de la simple expression du genre A=-10 à une expression du genre X=X+SIN(X)/(1+SQR(ABS(X*3))) par exemple, en passant par d'autres formes plus embrouillées.

Lorsque l'on n'est pas sûr de l'ordre dans lequel sera évaluée l'expression, on pourra toujours préciser cet ordre en ajoutant des parenthèses.

L'interpréteur Basic effectue les opérations entre parenthèses en procédant comme suit :

1. Il calcule chaque "sous-expression" entre parenthèses, en commençant par la sous-expression la plus intérieure.
2. A profondeur égale, il évalue la sous-expression située le plus à gauche dans l'expression.
3. Il exécute les opérations les plus prioritaires selon les règles de priorité du tableau suivant :

* OPERATEUR	:	NOTATION	PRIORITE *

* EXPONENTIATION	:	$\uparrow$	1 *
* NEGATION	:	-	2 *
* MULTIPLICATION & DIVISION	:	* /	3 *
* ADDITION ET SOUSTRACTION	:	+ -	4 *
* OPERATEURS RELATIONNELS	:	NOT AND OR	5 *

Exemple :

$$Y = A+B*\text{COS}(X)-(3*X\uparrow 2)/5$$

1. Evaluation de $(3*X\uparrow 2)$: il calcule les résultats intermédiaires qu'on appellera arbitrairement P1 et P2 :

$$P1 = X\uparrow 2$$

$$P2 = 3*P1$$

2. Evaluation de $B*\text{COS}(X)$ et les résultats intermédiaires P3 et P4 :

$$P3 = \text{COS}(X)$$

$$P4 = B*P3$$

3. Calcul de $(3*X\uparrow 2)/5$ à partir de P2 :

$$P5 = P2/5$$

4. Addition et soustraction :

$$A+P4-P5$$

Pour évaluer une expression ne contenant aucune parenthèse, et à priorité égale, on commence par les opérations situées le plus à gauche dans l'expression.

Pour en terminer sur ce point, reprenez cette simple règle : mettez entre parenthèses les opérations que vous voulez effectuer en premier lieu.

LES FONCTIONS STANDARDS DU BASIC II

X=SQR(12)

Dans cet exemple, SQR(..) représente une fonction standard du Basic II. Dans un programme, on peut faire référence à certaines fonctions connues du Basic, sans avoir à les définir au préalable. Ces fonctions, appelées fonctions standards peuvent se classer en quatre catégories :

- les fonctions mathématiques usuelles,
- les fonctions de conversion,
- les fonctions relatives au traitement des chaînes de caractères,
- les fonctions spéciales.

IMPORTANT :

Une fonction est toujours située à droite du symbole d'affectation. X=F(...) et non F(...)=...

Le résultat délivré par une fonction mathématique est toujours exprimé en simple précision.

Toutes les fonctions du Basic, à part quelques exceptions évidentes, sont récurives. C'est-à-dire qu'une fonction peut s'appeler elle-même plusieurs fois.

Un exemple de récursivité :

A=SQR(SQR(4))

selon la règle d'évaluation des expressions que nous avons vue plus haut, SQR(4) est d'abord calculé, ce qui donne 2 ; puis SQR(2) est évalué, ce qui donne comme résultat final : 1.41421

Les fonctions mathématiques usuelles*ABS(X)*

Donne la valeur absolue d'un nombre (signe toujours positif).

ABS(12.75)	donne	12.75
ABS(-65.3)	donne	65.3
ABS(-5.9)	donne	5.9

INT(X)

Donne la partie entière de X. Le plus grand entier inférieur ou égal à X ne se calcule pas de la même manière selon que X est négatif ou positif.

INT(12.5)	donne	12
INT(-3.5)	donne	-4
INT(.5)	donne	0
INT(-.5)	donne	-1

FIX(X)

Permet de récupérer la partie entière d'un nombre (troncature partie fractionnaire).

FIX(12.5)	donne	12
FIX(.5)	donne	0

Pour un nombre X non négatif, $FIX(X) = INT(X)$.

Par contre, pour un nombre négatif, $FIX(X) = INT(X) - 1$.

FIX(-3.5)	donne	-3
FIX(-.5)	donne	0

EXP(X)

Calcule e (base des logarithmes népériens) à la puissance X.

EXP(12.5)	donne	268337
EXP(-3.5)	donne	.0301974
EXP(.5)	donne	1.64872
EXP(1)	donne	2.71828
EXP(0)	donne	1

SQR(X)

Extrait la racine carrée de X.

SQR(12.34)	donne	3.51283
SQR(45)	donne	6.70821
SQR(123456)	donne	351.363
SQR(12E-5)	donne	.0109545

COS(X)

Retourne le cosinus de X exprimé en radians.

COS(60)	donne	-.952413
COS(45)	donne	.525322

SIN(X)

Retourne le sinus de X exprimé en radians.

SIN(45)	donne	.850902
SIN(60)	donne	-.304809

TAN(X)

Retourne la tangente de X exprimée en radians.

TAN(45)	donne	1.61977
TAN(60)	donne	.320039

ATN(X)

Retourne l'arc tangente de X (c'est à dire l'arc de cercle dont la tangente est X).

ATN(60)	donne	.320039
ATN(12E-5)	donne	.0109545

SGN(X)

Nous donne le signe de X.

SGN(12)	donne	1 (positif)
SGN(-12)	donne	-1 (négatif)
SGN(0)	donne	0

LOG(X)

Retourne le logarithme naturel (base e). L'expression X doit être positive.

LOG(2)	donne	.693147
LOG(.5)	donne	-.693147
LOG(EXP(1))	donne	1

NOTE IMPORTANTE : Toutes les fonctions précitées donnent un résultat en simple précision.

*Les fonctions de conversion**CDBL(X)*

Retourne la représentation en double précision de l'expression X.

CDBL(-12)	donne	-12
CDBL(-.4)	donne	-.4000000059604645
CDBL(-12E-9)	donne	-1.2000000099288445D-08

CINT(X)

Retourne la représentation entière de l'expression X, dans les limites : $-32768 \leq X \leq +32767$.

CINT(12.6)	donne	12
CINT(12E+3)	donne	12000
CINT(EXP(1))	donne	2

CSNG(X)

Retourne la représentation en simple précision de la valeur de l'expression X.

CSNG(12E-6)	donne	1.2E-05
CSNG(.2E-2)	donne	2E-03

RND(0)

Retourne un nombre pseudo-aléatoire compris entre 0.000001 et 0.999999.

RND(0)	donne	.36345
RND(0)	donne	.0710848

RND(X)

Retourne un nombre entier pseudo-aléatoire compris entre 1 et la valeur de l'expression X. (X) devant être inférieur à 32768.

RND(5)	donne	4
RND(5)	donne	2

Les fonctions sur chaînes de caractères

Les chaînes de caractères manipulées en Basic sont essentiellement des chaînes alphanumériques. Le codage utilisé est celui du standard (norme) ASCII. En annexe, vous trouverez un tableau représentatif des codes ASCII utilisables avec le Basic du TRS-80.

Rappelons qu'une constante ou une variable de chaîne peut avoir une longueur de 0 à 255 caractères.

ASC(C)

Retourne la valeur décimale de l'expression de chaîne C.

X=ASC("A")	donne	65 dans la variable X
B\$="W"		
X=ASC(B\$)	donne	87 dans la variable X

CHR\$(X)

Retourne, dans une variable de chaîne, la valeur de l'expression numérique X. (Inverse de la fonction ASC).

A\$=CHR\$(65)	donne "A" dans la variable A\$
X=87	
A\$=CHR\$(X)	donne "W" dans la variable A\$

LEN(C)

Retourne, dans une variable numérique, la longueur de l'expression de chaîne C.

L=LEN("TRS-80")	donne 6 dans la variable L
A\$="TRS-80 BASIC II"	
L=LEN(A\$)	donne 15 dans la variable L
B\$=""	(chaîne vide ou nulle)
L=LEN(B\$)	donne 0 dans la variable L
L=LEN(A\$+"RADIO-SHACK")	donne 27 dans la variable L

LEFT\$(C,X)

Prend la partie gauche de l'expression de chaîne C, sur une longueur égale à l'expression numérique X.

A\$="TRS-80"	
B\$=LEFT\$(A\$,3)	donne "TRS" dans la variable B\$
N=4	
B\$=LEFT\$(A\$,N)	donne "TRS-" dans la variable B\$
C\$=LEFT\$(A\$,N/2)	donne "TR" dans la variable C\$

RIGHT\$(C,X)

Prend la partie droite de l'expression de chaîne C, sur une longueur égale à l'expression numérique X.

A\$="TRS-80"	
B\$=RIGHT\$(A\$,3)	donne "80" dans la variable B\$
N=3	
C\$=RIGHT\$(A\$,N)	donne "-80" dans la variable C\$
D\$=RIGHT\$(A\$,N/2)	donne "0" dans la variable D\$

Notez ici que l'expression numérique qui définit la longueur de la sous-chaîne à extraire est entière. Ainsi, 3/2 donne 1.

Il en sera de même pour toutes les autres fonctions sur chaînes de caractères.

MID\$(C,X1,X2)

Extrait une sous-chaîne de l'expression de chaîne C. La sous-chaîne à extraire commence à la position exprimée par X1, et a pour longueur celle exprimée par X2. L'expression numérique X1 doit absolument être différente de zéro et inférieure à la longueur de la chaîne + X2.

X\$="TRS-80"

A\$=MID\$(X\$,4,1) donne "-" dans la variable A\$

A\$=MID\$(X\$,1,3) donne "TRS" dans la variable A\$
idem LEFT\$

A\$=MID\$(X\$,5) donne "80" dans la variable A\$

La longueur n'étant pas spécifiée, MID\$ prend tout le reste de la chaîne X\$, à partir de la position 5.

D=2

L=3

A\$=MID\$(X\$,D,L) donne "RS-" dans la variable A\$

Autre possibilité :

A\$=MID\$(LEFT\$(A\$,4),2,2)

est traitée comme suit :

1. extraction de la partie gauche de A\$: LEFT\$(A\$,4)="TRS-"
2. extraction de la sous-chaîne "RS" de la chaîne "TRS-" ainsi équivalent à MID\$("TRS-",2,2) soit la sous-chaîne "RS".

Une autre encore : Imbrication des fonctions !

X\$="LANGUAGE BASIC"

X\$=MID\$(MID\$(X\$,4,4),2,2)

qui retourne dans A\$ la sous-chaîne "AG" !!

STRING\$(X,C)

Crée une chaîne composée de X caractères C.

A\$=STRING\$(4,"*") donne "****" dans la variable A\$

I=4

B\$=STRING\$(I,"-") donne "----" dans la variable B\$

C\$="TRUC"

B\$=STRING\$(4,MID\$(C\$,3,1))

donne "UUUU" dans la variable B\$

STR\$(X)

Convertit l'expression numérique X en une chaîne décimale ASCII

X=123

A\$=STR\$(X) donne "123" dans la variable A\$

B\$=STR\$(12345) donne "12345" dans la variable B\$
identique à B\$="12345"

VAL(C)

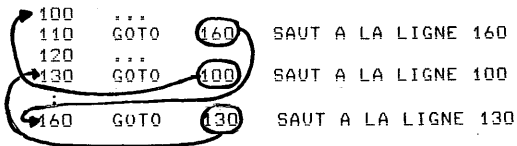
Retourne une valeur numérique entière correspondant à la valeur numérique de la chaîne C. C'est la fonction inverse de STR\$(X).

X=VAL("123") donne 123 dans la variable X
A\$="1234"
Y=VAL(A\$) donne 1234 dans la variable Y
Z=VAL(MID\$(A\$,2,2)) donne 23 dans la variable Z
U=VAL(RIGHT\$(A\$,3)) donne 234 dans la variable U

LES INSTRUCTIONS DE SAUT ET DE TEST

GOTO

(On dit branchement) car elle permet, comme son nom l'indique, de sauter d'une ligne de programme à une autre ligne de programme. Elle est toujours de la forme GOTO numéro de ligne.



Rien de plus simple que l'instruction GOTO !

200 GOTO 200

à votre avis, que se passe t-il ?

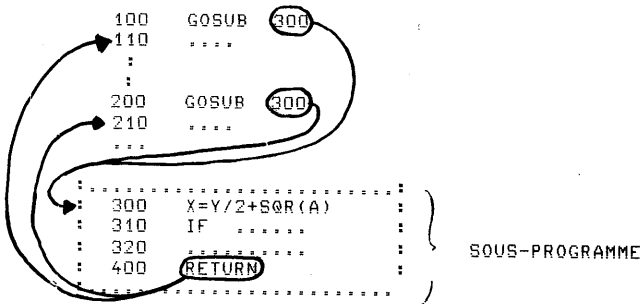
GOSUB et RETURN (sous-programme)

Par sous-programme, on entend une séquence d'instructions se terminant par l'instruction RETURN (retour).

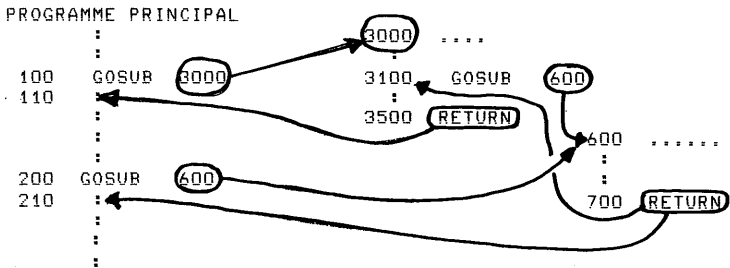
Cette méthode d'écriture permet d'éviter la duplication de séquences d'instructions identiques, ce qui offre l'avantage de diminuer l'encombrement mémoire du programme.

Le branchement vers un sous-programme se fait au moyen de l'instruction GOSUB, suivi du numéro de ligne de la prochaine instruction à exécuter.

L'instruction RETURN provoque le retour à l'instruction qui suit immédiatement l'instruction GOSUB ayant provoqué le branchement.



Quelques règles relatives aux sous-programmes.



- Avant de rencontrer une instruction RETURN, il faut au préalable avoir exécuté au moins une instruction GOSUB.
- Un sous-programme peut appeler lui-même un autre sous-programme et ainsi de suite. L'exemple ci-dessus montre que l'enchevêtrement des sous-programmes peut être parfois très compliqué.
- Il est déconseillé de sortir d'un sous-programme autrement que par une instruction RETURN.
- Contrairement aux fonctions que nous avons vues plus haut, les sous-programmes en Basic II ne sont, en principe, pas récursifs.
- Un sous-programme peut avoir plusieurs points d'entrée, à condition, toutefois, qu'il n'y ait pas d'entrave à la règle : "pas de RETURN sans GOSUB".

IF

Cette instruction permet de faire une comparaison entre deux expressions de même nature (arithmétiques, chaînes).

La forme générale de l'instruction IF est la suivante :

IF exp-1 OPERATEUR exp-2 MOT-CLE exp-3

exp-1 et exp-2 représentent les deux éléments à comparer.

Par exemple : A(2) avec B(3), A\$ avec "OUI"...

OPERATEUR désigne le type de comparaison à effectuer parmi les opérateurs suivants :

* OPERATEUR :	SIGNIFICATION	*

* = :	EGAL A	*
* < :	INFERIEUR A	*
* > :	SUPERIEUR A	*
* <= OU =< :	INFERIEUR OU EGAL A	*
* >= OU => :	SUPERIEUR OU EGAL A	*
* <> OU >< :	DIFFERENT DE	*

MOT-CLE spécifie si exp-3 doit être exécuté selon que le résultat de la comparaison est vrai ou faux. Deux mots-clés sont autorisés en Basic II :

THEN si le résultat est "vrai"
 ELSE si le résultat est "faux"

Exemples 1) IF A(2) = B(3) THEN...
 2) IF A <> 5 ELSE...

en clair 1) exécuter...si A(2) = B(3)
 2) ne pas exécuter...si A <> 5, autrement dit :
 exécuter...si A=5

exp-3 désigne, soit une expression à calculer, soit un numéro de ligne vers laquelle il faut aller pour continuer le programme (branchement), soit une instruction quelconque qui peut être aussi une instruction IF.

```

110 IF A(2) = B(3) THEN GOTO 200
120 .....
:
200 ..... SAUT ICI SI A(2) EST EGAL A B(3)
    
```

L'instruction IF est située à la ligne numéro 110 ; si A(2) = B(3), la prochaine instruction à exécuter se trouve à la ligne 200 (on exécute l'instruction GOTO 200).

Si la condition n'est pas satisfaite, on passe à l'exécution de l'instruction située à la ligne 120.

Utilisée sous cette forme, l'instruction GOTO peut être omise.

```
110 IF A(2) = B(3) THEN 200.
```

a le même effet.

Utilisation des opérateurs relationnels dans une instruction IF

Voilà une possibilité supplémentaire assez intéressante.
Supposons la séquence suivante :

```
>110 IF A(2) = B(3) THEN 200
>120 ....
>130 ....
>200 IF A(2) = 10 THEN 250
>210 GOTO 120
>220 ...
>250 ...
>_
```

qui consiste à tester $A(2)$ et $B(3) = 10$. Si c'est vrai, exécuter l'instruction située à la ligne 250. Si c'est faux, exécuter l'instruction située à la ligne 120. Pour effectuer ce test, on a employé deux instructions IF.

Avec l'utilisation d'opérateur relationnel, on écrirait :

```
110 IF (A(2) AND B(3)) = 10 THEN 250
120
:
:
250 .....
```

Autre exemple :

```
110 IF A(2) = 10 THEN 200
120 IF B(3) = 10 THEN 200
130 .....
:
:
200 .....
```

dans lequel on veut exécuter l'instruction située à la ligne 200 si $A(2)$ ou $B(3) = 10$. Avec l'opérateur relationnel OR, on écrirait:

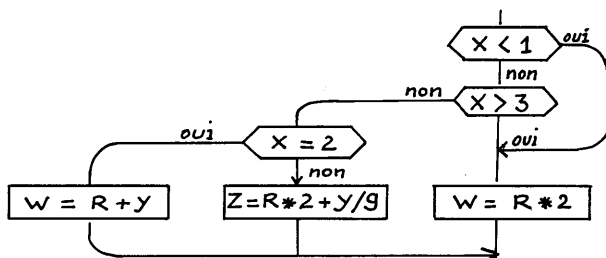
```
110 IF (A(2) OR B(3)) = 10 THEN 200
120 .....
200 .....
```

On peut aussi utiliser dans l'instruction IF plusieurs opérateurs relationnels, par exemple :

```
110 IF ((A(2) OR B(3)) AND (C(2) OR B(3))) = 10 THEN 200
```

ce qui signifie en clair : exécuter l'instruction située à la ligne 200 si $A(2)$ ou $B(3)$ et $C(2)$ ou $B(3) = 10$.

Résumons-nous dans l'organigramme ci-dessous :



La variable X ne pouvant prendre que les valeurs 1, 2, 3, la séquence d'instructions peut être la suivante :

```

100 IF X < 1 OR X > 3 THEN 140
110 IF X = 2 THEN 160
120 Z=R*2+Y/9
130 GOTO 170
140 W=R*2
150 GOTO 170
160 W=R*Y
170 ...
  
```

Voyons maintenant l'utilisation de IF avec des expressions de chaînes de caractères.

Tout d'abord, une définition à retenir : une chaîne de caractères A\$ est inférieure à une chaîne de caractères B\$ si, dans un classement alphabétique, la chaîne A\$ se trouve placée avant la chaîne B\$.

Exemple :

```

"B" < "C"
"AB" > "AA"
"TRS-79" < "TRS-80"
  
```

Les symboles < (plus petit que) ou > (plus grand que) sont les mêmes que ceux utilisés pour les variables numériques.

Exemple :

```

100 IF A$ = "OUI" THEN 200
100 IF LEFT$(A$,1) = "0" THEN 200
100 IF MID$(A$,3,1) <> "N" THEN 200
100 IF LEFT$(A$,3) = RIGHT$(B$,3) THEN 200
100 IF A$(1) < LEFT$(B$,LEN(B$)/2) THEN 200
100 IF A$+"TOTO" = B$ ELSE 200
  
```

Pour tester une chaîne vide, on peut programmer :

```

100 IF A$ = "" THEN 200
  
```

ou encore :

```

100 IF LEN(A$) = 0 THEN 200
  
```

Dans le premier cas, on compare la variable de chaîne A\$ avec une constante de chaîne vide ""; dans le deuxième cas, on teste si la longueur de la variable de chaîne A\$ vaut 0.

QUELQUES SUBTILITES A PROPOS DE L'INSTRUCTION IF

Une forme de test originale :

```
100 IF A THEN 200
```

effectue un branchement à la ligne 200 si la variable A est différente de zéro, ce qui évite d'écrire :

```
100 IF A <> 0 THEN 200
ou encore
100 IF A = 0 ELSE 200
```

On peut aussi écrire :

```
100 IF A ELSE 200
```

qui signifie aller en 200 si la variable A est nulle.

Et à propos des opérateurs relationnels, si l'on écrit :

```
100 A=10
110 B=15
120 C=A<B      (l'astuce !)
```

C vaut, dans ce cas, - 1 c'est-à-dire que la relation A<B est vraie.

Mais si l'on écrit :

```
120 C=A>B      (la même astuce inversée !)
```

C vaudra 0, car la relation A>B est fausse.

De même :

```
120 C = A<>B   (les deux combinées !)
```

donne C=1, car la relation est vraie, en effet la variable A est bien différente de la variable B.

La dernière :

```
120 C = A = B   (attention, par de confusion !)
```

C vaudra 0, car la relation A = B est fausse.

Méfiez-vous, car tous les Basics ne réagissent pas ainsi.

Par exemple :

```
100 A = B = C = 10
```

ce qui est tout-à-fait correct dans certains Basics, c'est à dire une affectation multiple très évidente (A=10, B=10, C=10) ; n'est pas possible avec Thérèse.

GOTO calculé

Dans un exemple précédent, on a effectué 3 tests pour déterminer trois traitements différents ($X=1$ ou 3 , $X=2$ et $X \in \{1,2,3\}$). L'instruction GOTO calculé résout plus facilement ce petit problème. Sa représentation est la suivante :

ON exp. GOTO L1,L2...L $\bar{n}$

ce qui veut dire ; en fonction de la valeur de l'expression exp. exécuter, soit l'instruction située à la ligne L1, soit l'instruction située à la ligne L2 etc...

Si l'évaluation de exp. est égale à zéro, il n'y a pas de branchement, on exécute dans ce cas l'instruction qui suit immédiatement le GOTO calculé.

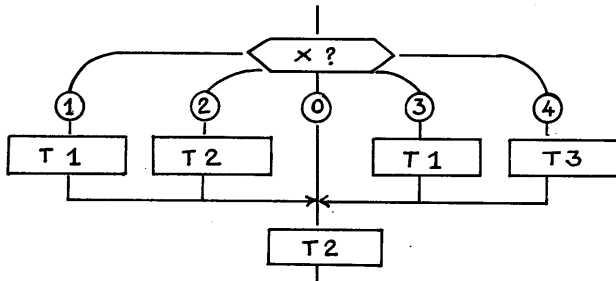
Avec l'utilisation de cette instruction, l'exemple précédent s'écrit :

```
100 ON X GOTO 110,130,150
110 W=R*2
120 GOTO 160
130 W=R+Y
140 GOTO 160
150 W=R*2+Y/9
160 ....
```

Si X vaut 1 on va à la ligne 110, si X vaut 2, on va à la ligne 130 et si X vaut 3, on va à la ligne 150. Dans les autres cas ($X=0$ ou $X > 3$) on continue à la ligne 110 ; très simple, n'est-ce pas ?

GOSUB calculé

De nombreux systèmes ne disposent pas de cette possibilité intéressante au même titre que le GOTO calculé. Illustrons ce cas par l'exemple ci-dessous :



T1, T2, T3, représentent certains traitements à effectuer selon la valeur de la variable X.

Cet organigramme se traduit normalement par la séquence suivante:

```

10      IF X=1 THEN 100
20      IF X=2 THEN 200
30      IF X=3 THEN 100
40      IF X=4 THEN 300
T2 [ 50      ...
    60      ...
T1 [ 100     ...
    150     GOTO 50
T2 [ 200     ...
    250     GOTO 50
T3 [ 300     ...
    350     GOTO 50

```

} Traitement répété deux fois

Avec utilisation du GOSUB calculé, on écrirait :

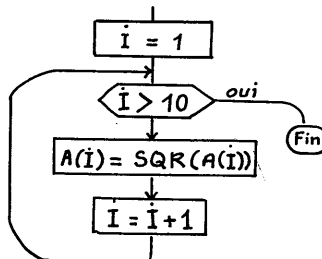
```

10      ON X GOSUB 100,200,300
50      GOSUB 200
60      ...
T1 [ 100     ...
    150     RETURN
T2 [ 200     ...
    250     RETURN
T3 [ 300     ...
    350     RETURN

```

BOUCLES DE PROGRAMME : Les instructions FOR et NEXT

Dans de nombreux problèmes, il est nécessaire de répéter plusieurs fois une série d'instructions. Considérons le problème suivant : Calcul de la racine carrée des 10 éléments d'un tableau. L'organigramme correspondant à ce problème est le suivant :



L'organigramme pourra se traiter de la façon suivante :

```

100 DIM A(10)
110 ...
120 INITIALISATION DU TABLEAU
130 ...
200 I=1                INDICE DE DEPART
210 IF I=10 THEN 300  FIN DE CALCUL
220 A(I)=SQR(A(I))    RACINE CARREE DE A(I) DANS A(I)
230 I=I+1             PROGRESSION DE L(INDICE
240 GOTO 210          ITERER
300 STOP              FIN DE CALCUL
    
```

On voit que l'on est conduit à itérer 10 fois la séquence d'instructions de la ligne 210 à la ligne 240, puis, les dix itérations terminées, aller à la ligne 300.

Ce genre de problème est résolu plus aisément avec les instructions d'itérations FOR et NEXT.

La forme générale de l'instruction FOR est la suivante :

FOR var = exp1 TO exp2 STEP exp3

var : désigne une variable de contrôle qui correspond à I dans l'exemple cité plus haut.

exp1 : désigne la valeur initiale à affecter à la variable de contrôle (1 dans l'exemple).

exp2 : désigne la valeur finale de la variable de contrôle, valeur qui conditionnera la fin d'exécution de la boucle (10 dans l'exemple).

STEP : est un mot réservé qui définit exp3 comme étant la valeur de l'incrément qui viendra s'ajouter à var.

L'incrément exp3 est facultatif, par défaut le pas STEP est égal à 1.

A l'instruction FOR est toujours associée une instruction NEXT qui a pour but de délimiter la séquence sur laquelle on doit itérer.

La forme de l'instruction NEXT est la suivante :

NEXT var

var : désigne la variable qui correspond à la variable de contrôle de l'instruction FOR associée.

Reprenons maintenant l'exemple avec utilisation de ces deux instructions :

```

100 DIM A(10)
110 ...
120 ...
150 ...
200 FOR I=1 TO 10
210 A(I) = SQRT(A(I))
230 NEXT I
300 <=== RETOUR ICI APRES 10 ITERATIONS

```

Exemple de calcul des 10 premiers termes de la série :
 $1/1 + 1/3 + 1/5 + 1/7 + \dots + 1/19$.

```

100 SERIE = 0
110 FOR I = 1 TO 19 STEP 2
120 SERIE = SERIE + 1/I
130 NEXT I

```

Le même exemple, mais en procédant à l'envers (utilisation d'un pas négatif) :

```

100 SERIE = 0
110 FOR I = 19 TO 1 STEP -2
120 SERIE = SERIE + 1/I
130 NEXT I

```

Autre exemple : calcul de la moyenne des éléments d'un tableau de 100 nombres :

```

100 DIM A(100)
110 ...
120 ..
200 MOYENNE = 0           Initialisation à 0
210 FOR I = 1 TO 100
220 MOYENNE = MOYENNE + A(I)  Somme des valeurs
230 NEXT I
300 MOYENNE = MOYENNE / 100  Résultat

```

Encore un exemple : calcul de PI à partir de la série
 $(PI+2)/6$

```

100 S=1
110 FOR I = 2 TO 40000
120   X=S+1/(I*I)
130   IF X=S THEN 170
140   S=X
150 NEXT I
160 I=40000
170 PRINT "I=";I,"PI=";SQRT(6*S)

```

Soyez patients, si vous voulez voir le résultat !!

Règles et remarques élémentaires relatives à une boucle FOR-NEXT.

- L'instruction NEXT doit être postérieure à l'instruction FOR associée.
- La variable de contrôle ne peut être une variable indicée.
- Un branchement de l'extérieur d'une boucle FOR-NEXT à l'intérieur d'une boucle est strictement interdit.
- On peut toujours quitter une boucle FOR-NEXT avant que la variable de contrôle ait atteint la valeur limite.
- Lorsque l'on quitte une boucle FOR-NEXT avant que la variable de contrôle ait atteint la valeur limite, la valeur de la variable de contrôle est égale à celle qu'elle avait juste avant de quitter la boucle.

Exemple :

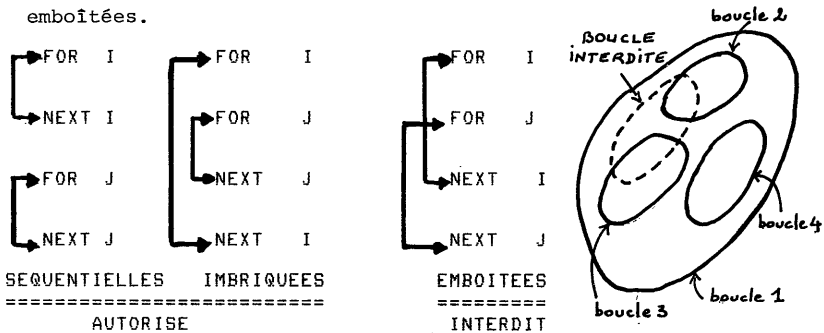
```

10  FOR I = 1 TO 100
20  ...
50  IF T(I) > M(I) THEN 200
60  ...
100 NEXT I
150 ...
200 ...

```

Si le test $T(I) > M(I)$ est satisfait lorsque I atteint, par exemple, la valeur 63, il y aura un saut à la ligne 200 et la valeur de I restera égale à 63.

- Lorsque l'on quitte une boucle FOR-NEXT par épuisement, la valeur de la variable de contrôle correspond à la dernière valeur possible, augmentée du pas.
- Un programme peut contenir plusieurs boucles FOR-NEXT imbriquées ou séquentielles, à condition qu'elle ne soient pas emboîtées.



Un petit conseil au passage :

Pour améliorer la lisibilité d'un programme, on peut décaler certaines instructions vers la droite, notamment les instructions de la portée d'une boucle FOR-NEXT.

Exemple :

```
100 A=10
110 FOR I = 1 TO 100
120     FOR J = 1 TO 10
130         A(I)=B(J)+A(I)
140     NEXT J
150 NEXT I
```

LES INSTRUCTIONS D'ENTREE/SORTIE

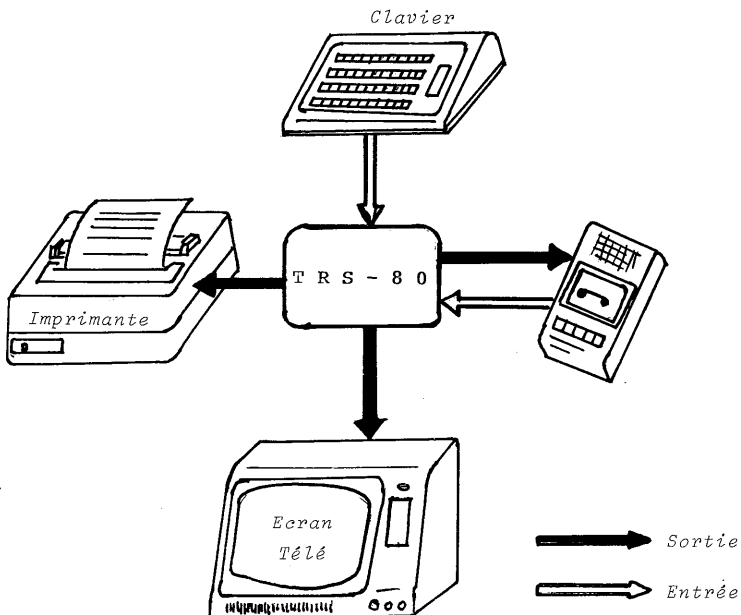
Lors de l'exécution d'un programme, le besoin d'échanges d'informations entre l'ordinateur et l'utilisateur se fait au moyen d'instructions appelées : "instructions d'Entrées/Sorties" (E/S en abrégé).

Le terme ENTREE au sens : UTILISATEUR → ORDINATEUR

Le terme SORTIE au sens : ORDINATEUR → UTILISATEUR

Les organes mis en oeuvre par ces instructions sont appelés "périphériques" au sens large du terme, tout simplement parce qu'ils se situent à l'extérieur de l'ordinateur, et font partie de son environnement.

Les périphériques sont souvent limités à un simple clavier de machine à écrire (organe d'entrée), d'un organe de sortie, comme par exemple un poste de télé, d'un organe E/S comme par exemple un magnétophone à cassette capable de lire (entrée) ou d'écrire (sortie) des données.



Généralités sur les instructions d'E/S élémentaires.

Ces instructions ont pour nom les mots réservés suivants :

READ	}	pour les instructions d'entrée
ou		
INPUT		
PRINT		pour l'instruction de sortie

Une opération d'entrée peut être effectuée à partir :

- de données préparées en mémoire dans des lignes numérotées (ce sont des données statiques). La lecture de ces données se fait essentiellement à partir de l'instruction READ ;
- de données que l'on introduit au cours de l'exécution du programme (ce sont des données dynamiques). Dans ce cas, on utilisera l'instruction INPUT.

Forme des instructions d'entrées.

READ	}	suivi d'une liste de variables simples ou d'éléments de tableaux
ou		
INPUT		

Exemple :

```

10 DIM A(10),B$(10)
.
.
50 READ M,W,A(I),B$(I)

```

Une liste est une suite d'identificateurs, séparés par une virgule. Les données lues sont effectuées aux variables selon l'ordre de la liste, mais ne doivent pas se présenter de n'importe quelle manière, c'est-à-dire qu'elles doivent être conformes à la nature des variables décrites dans la liste. Ainsi, pour l'exemple ci-dessus, la suite de données :

12	125.6	-3	"TRS-80"	est correcte
↓	↓	↓	↓	
M	W	A(I)	B\$(I)	

Par contre, la suite :

12	"TRS-80"	-3	125.6	est incorrecte
↓	↓			
M	W			

W étant une variable numérique, la donnée présentée étant une constante-chaîne, une erreur sera détectée par l'instruction READ et le programme interrompu.

DATA, pour définir les données

Les données statiques lues par READ sont définies dans un programme par l'utilisation du mot réservé : DATA, suivi d'une liste de valeurs, séparés par une virgule.

Exemple :

```

100 DATA 12,-5
110 DATA "TRS-80"
120 DATA 8,6,0
...
200 READ A
210 READ B
220 READ C
230 READ F,J,K

```

L'instruction READ A lit la valeur 12 dans la variable A

L'instruction READ B lit la valeur -5 dans la variable B

etc...

On s'aperçoit qu'après chaque READ, on est prêt à lire la donnée suivante dans la liste de DATA, de même que l'instruction READ située à la ligne 230 lit 8 dans la variable F, 6 dans la variable J et 0 dans la variable K.

Mais, si maintenant on exécute, par exemple :

```

240 READ M

```

nous essayons de lire une septième donnée qui ne figure pas dans la liste des DATA. Une anomalie est détectée par l'instruction READ, un message d'erreur est retourné à l'utilisateur et le programme sera interrompu.

RESTORE

Cette instruction permet de relire les données à partir de la première instruction DATA.

En effet, chaque fois qu'un READ est effectué, on progresse dans la liste des données, et le seul moyen de retourner au début, c'est de programmer l'instruction RESTORE.

Exemple :

```

100 DATA 10,6,5,9,-3
110 FOR I=1 TO 5
120 READ A(I)
130 NEXT I
...
200 RESTORE
210 GOTO 110

```

L'instruction RESTORE peut être programmée, même avant que la liste de DATA ne soit épuisée, son effet étant de repositionner le pointeur READ en début de la liste.

Les données statiques peuvent être situées n'importe où dans un programme, un GOTO sur une instruction DATA étant sans effet, dans l'exemple précédent on aurait pu écrire :

```
210 GOTO 100
```

Le résultat aurait été le même.

Pour faciliter la lecture "visuelle", et aussi pour ne pas ralentir inutilement l'exécution d'un programme, il est préférable d'écrire toutes les données statiques à la fin du programme.

INPUT, définition des données

L'instruction INPUT est destinée à lire des données en cours d'exécution du programme. Ce mode d'entrée est dit "dynamique" ou "conversationnel".

La forme générale de l'instruction INPUT est la suivante :

```
INPUT liste de variables
```

La liste de variables a la même structure que pour l'instruction READ, cependant, INPUT peut s'adresser à un organe d'entrée déterminé.

Le type d'organe d'entrée est précisé à droite du mot-clé INPUT. L'absence du type de périphérique implique une lecture à partir du clavier du TRS-80.

L'organe d'entrée autre que le clavier est défini par le caractère # suivi de - (tiret) et d'un numéro (1 ou 2) si l'entrée s'effectue à partir d'un lecteur de cassette.

Exemple :

```
INPUT #-1, A$, B$
```

```
INPUT #-2, B$, M
```

et suivi d'un numéro (1 à 4) si l'entrée s'effectue à partir d'un lecteur de mini-disquettes.

Exemple :

```
INPUT #1 A$
```

Exemple d'utilisation de INPUT : initialisation d'un tableau de 10 éléments :

```
100 DIM A(10)
110 FOR I=1 TO 10
120   INPUT A(I)
130 NEXT I
```

L'entrée est effectuée à partir du clavier du TRS-80, mais, pour chaque entrée, l'instruction INPUT affiche sur l'écran un ? (point d'interrogation) qui signifie "je suis prêt à recevoir la (ou les) donnée(s)". La fin du message est matérialisée par l'appui sur la touche ENTER.

Exemple :

```

      ? 1234.56 ENTER
      ↑      ↑
affiché par l'instruction  introduit par
      INPUT              l'utilisateur
```

Ce n'est qu'après l'appui sur la touche ENTER que la donnée frappée au clavier est chargée dans la variable A(I).

Dans le cas d'un INPUT sur une liste de variables, par exemple :

```
100 DIM A(10), B(10)
110 FOR I = 1 TO 10
120   INPUT A(I), B(I)
130 NEXT I
```

l'exécution donne :

```

      ? 1234.56,58 ENTER
      ↑   ↑   ↑
affiché par  A(I) B(I)
      INPUT
```

La présence de la virgule est obligatoire pour séparer les deux données.

La valeur 1234.56 sera chargée dans A(I) dès que la "," sera frappée la valeur 58 sera chargée dans B(I) dès que la touche ENTER sera frappée, de même que le programme continuera à la ligne 130 etc...

Comme pour l'instruction READ, la suite de données introduites à partir d'une instruction INPUT doit être compatible avec la nature des variables décrites dans la liste.

Une chaîne de caractères peut être introduite à partir de INPUT.

Exemple :

```
INPUT A$
```

La chaîne frappée au clavier, par exemple THERESE, sera introduite dans la variable A\$:

? THERESE

mais si la chaîne à introduire dans la variable A\$ doit comporter des caractères spéciaux, il faut l'encadrer par deux guillemets.

Exemple :

? "12,RUE"

Ici, le caractère spécial "," (virgule) fait partie de la chaîne et n'est pas considéré comme séparateur de données.

Exemple :

INPUT A\$,B\$

? 12,RUE

La variable A\$ contiendra la chaîne "12"

La variable B\$ contiendra la chaîne "RUE"

Rappelons que les guillemets ne font pas partie de la chaîne.

Correction des erreurs de frappe : ←

- par appui sur la touche ← pour annuler le dernier caractère introduit,

- par appui simultané sur les touches SHIFT ← pour annuler tout le champ frappé.

Messages d'erreurs :

- si la donnée introduite est incompatible avec le type de la variable qui la reçoit, le message REDO est affiché sur l'écran et un nouveau ? indique qu'il faut réintroduire la donnée.
- si l'on fournit plus de données que la liste n'en spécifie, le message EXTRAIGNORED est affiché : autrement dit, les données superflues sont ignorées.

Autre forme de l'instruction INPUT

Après tout, pourquoi ne pas faire précéder le ? qui est affiché par INPUT d'un message en clair ?

C'est tout-à-fait possible en écrivant :

INPUT "AGE";AGE

à l'exécution, on aura sur l'écran :

AGE ?

La donnée frappée sera introduite dans la variable AGE . On peut aussi écrire :

INPUT "NOM,PRENOM,AGE";NOM,PRE,AGE

A l'exécution, on aura :

```
NOM,PRENOM,AGE ? TOTO,LOUIS,18 ENTER
                   ↓   ↓   ↓
                   NOM PRE AGE
```

L'instruction de sortie PRINT ou LPRINT

La forme générale de cette instruction est la suivante :

PRINT liste

(LPRINT s'adresse à une imprimante, tandis que PRINT s'adresse à l'écran).

La liste peut contenir, non seulement des identificateurs de variable et des éléments de tableau, mais en plus des expressions arithmétiques et des éléments de mise en page.

Les éléments sont :

- les séparateurs : virgule (tabulation)
point-virgule
- les chaînes de caractères entre guillemets
- la fonction TAB qui sert à effectuer des tabulations.

Exemples :

```
PRINT A;"A";B2/I+J;"B2/I+J"
      ↑   ↑   ↑   ↑
variable chaîne expression chaîne
```

```
PRINT TAB(10),"*"
```

tabulation colonne 10 et impression d'un * puis retour à la ligne

- la fonction @ (A commercial) qui permet d'afficher un texte en n'importe quelle position de l'écran).
- la fonction USING qui permet d'éditer une donnée selon un format déterminé.

Une instruction PRINT peut s'écrire sans paramètres.

Exemple :

```
100 PRINT "TOTO"
110 PRINT
120 PRINT
130 PRINT "TITI"
```

Les PRINT des lignes 110 et 120 ont pour effet de créer deux interlignes supplémentaires entre la ligne où est imprimé TOTO et la ligne où est imprimé TITI.

La virgule

Outre sa fonction de séparateur de données, qu'elle remplit dans une liste, la virgule effectue une tabulation à une position déterminée dans la ligne.

Lorsqu'une virgule apparaît dans la liste, la donnée qui la précède est éditée en utilisant une zone de 16 caractères. Ainsi, la ligne d'écran qui contient 64 caractères comporte 4 zones tabulées comme ci-dessous :

```
1...16...32...48...64
```

```
100 PRINT "A","B","C","D","E"
```

donnera :

```
A   B   C   D   E
```

colonnes 1 16 32 48 64

Le point-virgule

Il joue également le rôle de séparateur de données dans une liste, mais, dans ce cas, aucune tabulation n'est effectuée après la donnée qui précède le point-virgule.

Exemple :

```
100 PRINT "A";"B";"C"
```

donnera ABC

Mais, dans le cas suivant :

```
100 A=10
```

```
110 B=20
```

```
150 PRINT "A=";A;"B=";B
```

donnera A= 10 B=20

Une donnée numérique est précédée, dans ce cas, par un caractère espace, emplacement réservé au signe.

```
100 A=-20
```

```
110 PRINT"A=";A
```

donne A=-20

Lorsque la donnée est positive, le signe "+" est omis. Il est remplacé par un espace (ou blanc). Cette remarque est également valable dans le cas de la virgule.

Lorsqu'un point-virgule apparaît à la fin d'une ligne PRINT, aucun interligne automatique n'est effectué.

Exemple :

```
100 PRINT "LE ";
110 PRINT "TRS-80 ";
120 PRINT "DE RADIO-SHACK"
```

} empêche de passer à la ligne suivante

donne pour résultat :

```
LE TRS-80 DE RADIO-SHACK
```

La fonction TAB

TAB(exp) peut apparaître dans une instruction PRINT. Cette fonction indique la position du curseur sur l'écran, (ou de la tête d'impression sur l'imprimante) où l'on doit imprimer la donnée.

Exemples :

```
110 PRINT TAB(10), "A";
```

affichera A en colonne dix de l'écran

```
100 FOR I=1 TO 10
110 PRINT TAB(I), "X";
120 NEXT I
```

donne pour résultat :

```
*****
1      10
TAB(1)  TAB(10)
```

Si I n'est pas un entier, TAB(I) est interprété de la même façon que TAB(INT(I)). Cette possibilité de tabulation permet d'améliorer la présentation des résultats, de faciliter l'édition des tableaux, de tracer des courbes d'une façon relativement simple.

La fonction @

Cette fonction permet d'atteindre n'importe quelle partie de l'écran. Elle est de la forme :

```
PRINT @ exp.
```

ou exp. désigne une adresse de l'écran (0 à 1023).

0 donne l'adresse de la première position de la première ligne, 63 celle de la dernière position de la première ligne... et 1023 la dernière position de la dernière ligne.

Cette fonction ne peut apparaître qu'une fois dans une ligne PRINT, et ce immédiatement après l'instruction.

Exemple :

```
PRINT@142,"A"
```

affiche la lettre A dans la case 142 de l'écran.

Le petit programme suivant permet d'afficher tous les codes compris entre 32 et 255, par exemple :

```
100 FOR I = 32 TO 255
110 PRINT@I-32;CHR$(I)
120 NEXT I
```

à partir de la case 0.

La fonction USING

Elle permet l'édition d'une donnée ou d'une liste selon un modèle d'édition déterminé.

Le format est le suivant :

```
PRINT USING modèle;liste
```

modèle est, soit une constante-chaîne (entre guillemets), soit contenu dans une variable de chaîne.

La chaîne constituant le modèle d'édition est composée d'un certain nombre de caractères spéciaux :

- # . spécifie la position de chaque chiffre. Les positions non utilisées à gauche (zéros non significatifs) sont remplacés par des espaces ; et celles de droite (après le point décimal) par des zéros.
- ** . deux astérisques au début du champ feront que les positions de gauche non significatives seront remplacées par des *
- \$\$. un caractère \$ sera imprimé devant le premier chiffre significatif.
- **\$. insertion de caractères * dans les positions de gauche non significatives plus un \$ devant le premier caractère significatif.
- + . placé au début ou à la fin du champ, permet l'édition du signe (+ ou -) à droite ou à gauche de la donnée, selon qu'elle est positive ou négative.
- . (moins) placé au début ou à la fin du champ, permet l'édition du signe "-" si la donnée est négative. Sinon, un espace est inséré à la place du signe.

% % . spécifie le nombre de caractères qui seront imprimés + 2
(on compte les %, plus les espaces entre les %).

! . indique qu'il faut extraire le premier caractère de la chaîne.

Pour illustrer cette fonction, voyons quelques exemples ci-dessous : (Lors de l'écriture de ce livre, nous avons essayé ces différents exemples. Dans certains cas, la réponse du TRS-80 n'a pas été la réponse attendue, que nous avons indiquée à droite de la réponse donnée) :

```
10 INPUT "FORMAT":F$
20 INPUT "DONNEE":D
30 PRINT "RESULTAT":
40 PRINT USING F$:D
50 GOTO 10
```

FORMAT	##.##
DONNEE	12.56
RESULTAT	12.56

FORMAT	##.##
DONNEE	123.56
RESULTAT	%123.56

FORMAT	##.##
DONNEE	12.589
RESULTAT	12.59

FORMAT	+##.##
DONNEE	12.56
RESULTAT	+12.56

FORMAT	##.##+
DONNEE	12.56
RESULTAT	12.56+

FORMAT	-##.##	Attendu :
DONNEE	-12.56	
RESULTAT	-%12.56	-12.56

FORMAT	##.##-
DONNEE	-12.56
RESULTAT	12.56-

FORMAT	.##.##-	Attendu :
DONNEE	12.56	
RESULTAT	12.56	%12.56

FORMAT	**##	
DONNEE	12.13	
RESULTAT	**12	
FORMAT	+***##.##	Attendu :
DONNEE	8	
RESULTAT	**+8*	+*** 8.00
FORMAT	\$##.##	
DONNEE	12.56	
RESULTAT	\$12.56	
FORMAT	\$\$\$.\$\$+	Attendu :
DONNEE	43.5	
RESULTAT	%\$44	\$43.50+
FORMAT	\$#,###.##	
DONNEE	1234.56	
RESULTAT	\$1,234.56	
FORMAT	\$#,###.##	
DONNEE	12.56	
RESULTAT	\$ 12.56	
FORMAT	!	
DONNEE	ABCD	
RESULTAT	A	
FORMAT	%%	
DONNEE	ABCD	
RESULTAT	AB	
FORMAT	%%%	
DONNEE	ABCDEFG	
RESULTAT	AB	
FORMAT	% %	
DONNEE	ABCDEFGHIJK	
RESULTAT	ABCDEF	

```
10 INPUT A$,B$,C$
20 PRINT USING "!":A$;B$;C$
```

```
ABCD,EFGH,IJKL
RESULTAT
AEI
```

EN REMPLACANT LA LIGNE 20 PAR :

```
20 LPRINT USING "!!":A$;B$;C$
```

AVEC LES MEMES DONNEES ON OBTIENT

```
A E I
```

LES FONCTIONS GRAPHIQUES

L'écran du TRS-80 géré en mode "graphique" comporte 48 lignes et 128 colonnes.

Un point de l'écran est adressé par ses coordonnées :

X = N° de colonne 0 à 127

Y = N° de ligne 0 à 47

La fonction SET (x,y)

Permet "l'allumage" d'un point situé aux coordonnées x, y.

Exemples :

```
100 SET(12,26)
```

allume un point situé à l'intersection de la colonne 12 et de la ligne 26 ;

```
100 SET(RND(128)-1,RND(48)-1)
```

allume un point, quelque part dans l'écran, x et y étant aléatoires.

La fonction RESET (x,y)

C'est l'inverse de la précédente, c'est-à-dire qu'elle permet l'extinction d'un point.

Exemple :

```
100 CLS
110 X=10:Y=20
120 SET(X,Y)          allumage
...
150 RESET(X,Y)        extinction
```

La fonction POINT (x,y)

Permet de tester l'état d'un point de l'écran. Un point est soit "allumé", soit "éteint".

Exemple :

```
100 SET(20,20)
110 X = POINT(20,20)
```

retourne la valeur -1 dans la variable X, ce qui correspond au point "allumé" ;

```
120 RESET(20,20)
130 X=POINT(20,20)
```

retourne la valeur 0 dans la variable X, ce qui correspond au point "éteint".

On peut ainsi écrire :

```
110 IF POINT(X,Y) THEN 1000
...
1000 PRINT "POINT";Y,Y;"ALLUME";
```

LES FONCTIONS SPECIALES

Outre les fonctions mathématiques et les fonctions sur chaînes de caractères déjà abondantes, le Basic II comporte un lot de fonctions spéciales assez intéressantes :

La fonction CLS

Efface l'écran, et positionne le curseur en haut et à gauche.

La fonction CLEAR n

Rend nulles les variables numériques, et vides les variables de chaîne et, si n est mentionné, elle réserve un espace de n octets pour le stockage des chaînes de caractères. Par défaut n est égal à 50.

Exemple CLEAR

CLEAR 2000

La fonction ERROR code

Permet de signaler une erreur.

Exemple :

```
100 IF A = 0 THEN ERROR 20
```

Si la variable A vaut 0, le message "UE ERROR" sera affiché sur l'écran, et le programme interrompu.

On peut aussi utiliser le mot-clé ERROR pour "récupérer" une erreur au lieu et place de l'arrêt du programme. Ainsi :

```
100 ON ERROR GOTO 1000
```

```
...
```

```
200 X=X/O (erreur volontaire -division par zéro-)
```

```
210
```

```
1000 ...
```

Le programme ne sera pas interrompu, et un branchement à la ligne 1000 sera effectué.

La fonction INKEY\$

Qui s'écrit : A\$=INKEY\$ par exemple.

permet de ranger dans la variable de chaîne A\$ un caractère (code d'une touche du clavier) lu "à la volée".

Contrairement à l'instruction INPUT qui attend patiemment la frappe d'un message terminé par ENTER, la fonction INKEY\$ n'effectue aucune attente et ne reçoit qu'un seul caractère.

Si une touche est enfoncée au moment où l'instruction est exécutée, le code du caractère correspondant est chargé dans la variable de chaîne (ici A\$). Si aucune touche n'est enfoncée au moment de l'exécution de la fonction, la variable de chaîne sera vide (longueur nulle).

Exemple :

```
10 A$=INKEY$
20 IF A$="" THEN 10
30 PRINT A$;
40 GOTO 10
```

Explication :

Ligne 10 : lecture du clavier et chargement de la variable A\$

Ligne 20 : teste si un caractère a été lu (A\$ non vide)

Si A\$ est vide, on retourne en 10 pour une nouvelle lecture.

Ligne 30 : impression du caractère lu.

Ligne 40 : retour en 10 pour une nouvelle lecture.

Il faut noter que la fonction INKEY\$ n'assure pas l'impression du caractère introduit au clavier ; la réception est dite "transparente", le curseur n'est également pas affiché.

Autre exemple : entrée d'un message de 4 caractères de long et affichage de celui-ci :

```
100 M$=""
110 C$=INKEY$
120 IF C$="" THEN 110
130 M$=M$+C$
140 IF LEN(M$)=4 ELSE 110
150 PRINT M$
160 GOTO 100
```

Explication :

Ligne 100 : la variable M\$ est rendue "VIDE". Elle contiendra le texte à afficher.

Ligne 110-120 : acquisition d'un caractère dans la variable C\$

Ligne 130 : le caractère courant (C\$) est ajouté (concaténation) à la variable M\$.

Ligne 140 : On itère tant que la variable M\$ ne contient pas 4 caractères. (Pourrait s'écrire 140 IF LEN(M\$) <> 4 THEN 110).

Ligne 150-160 : Impression de la variable M\$ et retour en 100.

La fonction RESUME numéro de ligne

Elle termine une séquence de traitement d'erreur. Le numéro de ligne spécifie la ligne de la prochaine instruction à exécuter.

Exemple :

```
100 ON ERROR GOTO 1000
...
200 X=X/0
210 PRINT"RETOUR/ERREUR"
...
1000 PRINT "ERREUR LIGNE";ERL
1010 RESUME 210
```

La fonction ERL

Retourne le numéro de ligne où une erreur a été détectée. Ne peut être exploitée que conjointement avec ON ERROR.

Exemple :

```
100 ON ERROR GOTO 1000
...
1000 X=ERL          (X contiendra le N° de ligne erroné)
1010 RESUME 100
```

La fonction ERR

Retourne une valeur liée au type de l'erreur détectée (voir liste des codes erreurs en annexe).

Exemple :

```
100 ON ERROR GOTO 1000
...
1000 PRINT "ERR";ERR/2+1;"LIGNE";ERL
1010 RESUME 100
```

où X = ERR valeur de l'erreur
X = ERR/2+1 code exact de l'erreur

La fonction MEM

Fournit le nombre d'octets inutilisés (libres) en mémoire.

Exemple :

```
X = MEM
PRINT "IL RESTE";MEM;"OCTETS EN MEMOIRE"
```

La fonction POS (0)

Retourne une valeur numérique entière, indiquant la position du curseur. L'argument (0) est une variable fictive. On écrit : X = POS(0). La variable X contiendra la position courante du curseur (0 à 63)

L'instruction STOP

S'écrit 100 STOP

ou 200 IF....THEN STOP

Elle entraîne l'arrêt du programme après affichage sur s'écran du message BREAK AT nnnn, où nn représente le numéro de ligne où a été rencontrée l'instruction STOP.

La poursuite de l'exécution du programme est effectuée après introduction, au clavier, du message CONTINUE qui signifie tout simplement continuer.

L'instruction END

Elle termine normalement un programme. L'exécution de cette instruction entraîne l'arrêt du programme et l'affichage du message READY.

Les fonctions spéciales ci-dessous, étant étroitement liées avec la programmation en langage assembleur du Z-80, seront développées en détail dans le deuxième volume. Il s'agit de :

- PEEK : lecture un octet en mémoire ;
- POKE : écriture un octet en mémoire ;
- DEFUSR : définition de l'adresse d'un sous-programme écrit en langage assembleur ;
- VARPTR : récupérer l'adresse d'une variable ;
- USR : appel d'un sous-programme écrit en langage assembleur ;
- INP : entrée d'une donnée d'un coupleur d'E/S ;
- OUT : sortie d'une donnée d'un coupleur d'E/S.

TOUCHE EST HORIZONTAL OU VERTICAL, AUQUEL CAS, IL FAUDRA INVERSER LE SIGNE DE L'INCREMENT DE DEPLACEMENT CORRESPONDANT (1 DEVIENT -1, -1 DEVIENT 1, CE QUI EST TOUT SIMPLEMENT REALISE EN MULTIPLIANT CET INCREMENT PAR LA VALEUR -1). L'IMPACT DE LA BALLE EST MATERIALISE VISUELLEMENT (LIGNE 270) ET AUDITIVEMENT (LIGNES 290 ET 300). CES DEUX DERNIERES COMMANDES VONT EN EFFET EXCITER LE 'RELAIS' QUI TELECOMMANDE ORDINAIREMENT LE MOTEUR DU MAGNETOPHONE, CE QUI PROVOQUE UN LEGER BRUIT... LE PROGRAMME SE POURSUIT ALORS EN LIGNE 190, NON SANS AVOIR PREALABLEMENT EFFACE LE MESSAGE "POC!".

DANS LE CAS OU LA BALLE N'A PAS ATTEINT L'UN DES BORDS, ELLE EST DESSINEE EN LIGNE 130. APRES UNE ATTENTE SUFFISANTE (LIGNE 150), ELLE EST EFFACEE (170) ET LE PROCHAIN DEPLACEMENT EST ALORS CALCULE AUX LIGNES 190 ET 200. LE PROGRAMME BOUCLE ALORS SUR LA LIGNE 110.

A N N E X E I

Caractéristiques du BASIC II

Données :

Entières : -32768 à + 32767 inclus
Simple précision: -1.701411E+38 à +1.701411E+38 inclus
Double précision: -1.701411834544556E+38 à
 +1.701411834544556E+38 inclus
Chaîne : 0 à 255 caractères

Numéro de ligne : 0 à 65529 inclus

Longueur d'une ligne de programme : jusqu'à 255 caractères.

Occupation mémoire : Chaque ligne de programme occupe au minimum 5 octets en mémoire, à savoir :

N° de ligne : 2 octets
Pointeur ligne : 2 octets
Retour chariot : 1 octet

De plus, chaque mot réservé, caractère spécial, opérateur etc... occupe 1 octet. La place occupée par un nom de variable dépend de la longueur de ce nom.

L'allocation des variables étant dynamique, on a :

Pour un entier : 5 octets (2 pour la valeur, 3 pour le nom de la variable).

Pour une variable simple précision : 7 octets (4 pour la valeur, 3 pour le nom).

Pour une variable double précision : 11 octets (8 pour la valeur, 3 pour le nom).

Pour une variable de chaîne : 6 octets minimum (3 pour le nom, 1 pour la longueur, 2 pour l'adresse de la chaîne + 1 octet par caractère de la chaîne).

Pour les tableaux : 12 octets minimum (3 pour le nom du tableau, 2 pour la taille, 1 par nombre de dimensions, 2 pour chaque dimension, et 2, 3, 4 ou 8 (selon le type de tableau) pour chaque élément dans le tableau).

Chaque boucle FOR-NEXT : nécessite 16 octets supplémentaires (en dehors du programme).

Chaque GOSUB actif : nécessite 6 octets (en dehors du programme).

Chaque niveau de parenthèses dans une expression : nécessite 4 octets plus 12 octets, pour stocker un résultat intermédiaire.

A N N E X E I I

Liste des codes erreurs BASIC II

Code :	Abréviation :	Signification :
1	NF	NEXT en dehors d'un FOR (imbrication boucles)
2	SN	Erreur syntaxe
3	RG	RETURN sans GOSUB
4	OD	Débordement de données (chaîne)
5	FC	Fonction illégale
6	OV	Débordement opération arithmétique
7	OM	Débordement mémoire (plus de place)
8	UL	Numéro de ligne non défini
9	BS	Sous-chaîne en dehors des limites de la chaîne
10	DD	Redimensionnement d'un tableau interdit
11	/0	Division par zéro
12	ID	Erreur dans instruction INPUT Exemple : INPUT 3
13	TM	Mélange de type de données
14	OS	Plus d'espace pour les chaînes
15	LS	Chaîne trop longue (255 caractères)
16	ST	Expression de chaîne trop complexe
17	CN	CONTINUE interdit après END ou EDIT
18	NR	Pas de RESUME
19	RW	Instruction RESUME rencontrée avant ON ERROR GOTO
20	UE	Erreur non imprimable
21	MO	Mélange d'opérations
22	FD	Donnée incorrecte
23	L3	Commande réservée au DISK BASIC

A N N E X E I I I

Mots réservés du BASIC II

ABS	FIX	OUT
AND	FOR	PEEK
ASC	FRE	POINT
ATN	* GET	POKE
CDBL	GOSUB	POS
CHR\$	GOTO	PRINT
CINT	IF	* PUT
CLEAR	INKEY\$	RANDOM
* CLOSE	INP	READ
CLS	INPUT	REM
* CMD	* INSTR	RESET
CONT	INT	RESTORE
COS	* KILL	RESUME
CSNG	LEFT\$	RETURN
* CVD	LET	RIGHT\$
* CVI	* LSET	RND
* CVS	LEN	* SAVE
DATA	* LINE	SET
DEFDBL	LIST	SGN
* DEFFN	* LOAD	SIN
DEFINT	* LOC	SQR
DEFSNG	* LOF	STEP
* DEFUSR	LOG	STOP
DEFSTR	MEM	STRING\$
DELETE	* MERGE	STR\$
DIM	* MID\$	TAB
EDIT	* MKD\$	TAN
ELSE	* MKI\$	THEN
END	* MKS\$	* TIME\$
ERL	* NAME	TROFF
ERR	NEW	TRON
ERROR	NEXT	USING
EXP	NOT	USR
* FIELD	ON	VAL
	* OPEN	VARPTR

* Ces mots réservés sont exploités par le Basic DOS.

ANNEXE I V

Les codes clavier et graphiques

Une chose n'est pas claire dans les manuels Tandy : Il n'y a pas de distinction entre les codes générés par le clavier, ceux générés par le moniteur, et ceux affectés aux graphismes et aux fonctions de la vidéo.

Nous donnons, ci-après, les trois tableaux correspondants.

Codes clavier *		NON-SHIFT				**		SHIFT				*
* TOUCHE *	* GRAPH *	* CODE *	* CODE *	* CODE *	* CODE *	** GRAPH *	* CODE *	* CODE *	* CODE *	* CODE *	*	
* CLAVIER *	* /ACTION *	* DECI *	* HEXA *	* DECI *	* HEXA *	** /ACTION *	* DECI *	* DECI *	* DECI *	* HEXA *	*	
* 1 *	* 1 *	* 49 *	* 31 *	** ! *	* 33 *	* 21 *	*	*	*	*	*	
* 2 *	* 2 *	* 50 *	* 32 *	** " *	* 34 *	* 22 *	*	*	*	*	*	
* 3 *	* 3 *	* 51 *	* 33 *	** # *	* 35 *	* 23 *	*	*	*	*	*	
* 4 *	* 4 *	* 52 *	* 34 *	** \$ *	* 36 *	* 24 *	*	*	*	*	*	
* 5 *	* 5 *	* 53 *	* 35 *	** % *	* 37 *	* 25 *	*	*	*	*	*	
* 6 *	* 6 *	* 54 *	* 36 *	** & *	* 38 *	* 26 *	*	*	*	*	*	
* 7 *	* 7 *	* 55 *	* 37 *	** ' *	* 39 *	* 27 *	*	*	*	*	*	
* 8 *	* 8 *	* 56 *	* 38 *	** (*	* 40 *	* 28 *	*	*	*	*	*	
* 9 *	* 9 *	* 57 *	* 39 *	**) *	* 41 *	* 29 *	*	*	*	*	*	
* 0 *	* 0 *	* 48 *	* 30 *	** SPACE *	* 32 *	* 20 *	*	*	*	*	*	
* : *	* : *	* 58 *	* 3A *	** * *	* 42 *	* 2A *	*	*	*	*	*	
* - *	* - *	* 45 *	* 2D *	** = *	* 61 *	* 3D *	*	*	*	*	*	
* BREAK *		* 01 *	* 01 *	** * *	* 01 *	* 01 *	*	*	*	*	*	
* ↑ *	* ↑ *	* 91 *	* 5B *	** CURS ↑ *	* 27 *	* 1B *	*	*	*	*	*	
* ↓ *	* CR+LF *	* 10 *	* 0A *	** CURS ↓ *	* 26 *	* 1A *	*	*	*	*	*	
* ← *	* BACKSP *	* 08 *	* 08 *	** CURS ← *	* 24 *	* 18 *	*	*	*	*	*	
* → *	* TABUL H *	* 09 *	* 09 *	** CURS → *	* 25 *	* 19 *	*	*	*	*	*	
* CLEAR *		* 31 *	* 1F *	** * *	* 31 *	* 1F *	*	*	*	*	*	
* A *	* A *	* 65 *	* 41 *	** A *	* 97 *	* 61 *	*	*	*	*	*	
* B *	* B *	* 66 *	* 42 *	** B *	* 98 *	* 62 *	*	*	*	*	*	
* C *	* C *	* 67 *	* 43 *	** C *	* 99 *	* 63 *	*	*	*	*	*	
* D *	* D *	* 68 *	* 44 *	** D *	* 100 *	* 64 *	*	*	*	*	*	
* E *	* E *	* 69 *	* 45 *	** E *	* 101 *	* 65 *	*	*	*	*	*	
* F *	* F *	* 70 *	* 46 *	** F *	* 102 *	* 66 *	*	*	*	*	*	
* G *	* G *	* 71 *	* 47 *	** G *	* 103 *	* 67 *	*	*	*	*	*	
* H *	* H *	* 72 *	* 48 *	** H *	* 104 *	* 68 *	*	*	*	*	*	
* I *	* I *	* 73 *	* 49 *	** I *	* 105 *	* 69 *	*	*	*	*	*	
* J *	* J *	* 74 *	* 4A *	** J *	* 106 *	* 6A *	*	*	*	*	*	
* K *	* K *	* 75 *	* 4B *	** K *	* 107 *	* 6B *	*	*	*	*	*	
* L *	* L *	* 76 *	* 4C *	** L *	* 108 *	* 6C *	*	*	*	*	*	
* M *	* M *	* 77 *	* 4D *	** M *	* 109 *	* 6D *	*	*	*	*	*	
* N *	* N *	* 78 *	* 4E *	** N *	* 110 *	* 6E *	*	*	*	*	*	
* O *	* O *	* 79 *	* 4F *	** O *	* 111 *	* 6F *	*	*	*	*	*	
* P *	* P *	* 80 *	* 50 *	** P *	* 112 *	* 70 *	*	*	*	*	*	
* Q *	* Q *	* 81 *	* 51 *	** Q *	* 113 *	* 71 *	*	*	*	*	*	
* R *	* R *	* 82 *	* 52 *	** R *	* 114 *	* 72 *	*	*	*	*	*	
* S *	* S *	* 83 *	* 53 *	** S *	* 115 *	* 73 *	*	*	*	*	*	
* T *	* T *	* 84 *	* 54 *	** T *	* 116 *	* 74 *	*	*	*	*	*	
* U *	* U *	* 85 *	* 55 *	** U *	* 117 *	* 75 *	*	*	*	*	*	
* V *	* V *	* 86 *	* 56 *	** V *	* 118 *	* 76 *	*	*	*	*	*	
* W *	* W *	* 87 *	* 57 *	** W *	* 119 *	* 77 *	*	*	*	*	*	
* X *	* X *	* 88 *	* 58 *	** X *	* 120 *	* 78 *	*	*	*	*	*	

Codes clavier (Suite)

* NON-SHIFT				** SHIFT			
* TOUCHE *	GRAPH	* CODE *	CODE	** GRAPH	* CODE *	CODE *	
* CLAVIER *	/ACTION	* DECI *	HEXA	** /ACTION	* DECI *	HEXA *	
* Y *	Y	* 89 *	59	** Y	* 121 *	79 *	
* Z *	Z	* 90 *	5A	** Z	* 122 *	7A *	
* , *	,	* 44 *	2C	** <	* 60 *	3C *	
* . *	.	* 46 *	2E	** >	* 62 *	3E *	
* SPACE *	SPACE	* 32 *	20	** SPACE	* 32 *	20 *	
* ; *	;	* 59 *	3B	** +	* 43 *	2B *	
* @ *	@	* 64 *	40	** @	* 96 *	60 *	
* ENTER *	CR+LF	* 13 *	0D	** CR+LF	* 13 *	0D *	
* / *	/	* 47 *	2F	** ?	* 63 *	3F *	

Codes fonctions moniteur

* TOUCHES *	ACTION	
* CLAVIER *	ET CODES ENGENDRES	
* CLEAR	* EFFACEMENT ECRAN	
	* (CODE HOME + EFFACEMENT FIN PAGE)	
* SHIFT →	* 32 CARACTERES PAR LIGNE	
* SHIFT @	* PAUSE	
* BREAK	* RETOUR A "READY >"	
* SHIFT ←	* EFFACEMENT LIGNE + RETOUR DEBUT	
	* DE LIGNE	

Codes vidéo engendrés par programme (CHR\$)

* CODE *	CODE *	ACTION OU GRAPHISME	
* DECI *	HEXA *		
* 14 *	0E *	* CURSEUR PRESENT (ON)	
* 15 *	0F *	* CURSEUR ABSENT (OFF)	
* 28 *	1C *	* HOME (DEBUT D'ECRAN)	
* 29 *	1D *	* CURSEUR EN DEBUT DE LIGNE	
* 30 *	1E *	* EFFACEMENT FIN DE LIGNE	
* 31 *	1F *	* EFFACEMENT FIN DE PAGE	
* 95 *	5F *	* CODE CURSEUR	
* 23 *	17 *	* MODE 32 CARACTERES PAR LIGNE	
* 92 *	5C *	* ↓	
* 93 *	5D *	* ←	
* 94 *	5E *	* →	

Les codes graphiques

128 (decimal) 80 (hexa)	129 81	130 82	131 83	132 84
133 85	134 86	135 87	136 88	137 89
138 8A	139 8B	140 8C	141 8D	142 8E
143 8F	144 90	145 91	146 92	147 93
148 94	149 95	150 96	151 97	152 98
153 99	154 9A	155 9B	156 9C	157 9D
158 9E	159 9F	160 A0	161 A1	162 A2
163 A3	164 A4	165 A5	166 A6	167 A7
168 A8	169 A9	170 AA	171 AB	172 AC
173 AD	174 AE	175 AF	176 B0	177 B1
178 B2	179 B3	180 B4	181 B5	182 B6
183 B7	184 B8	185 B9	186 BA	187 BB
188 BC	189 BD	190 BE	191 BF	

Nota : les zones noires et blanches sont inversées sur l'écran.

A N N E X E V

Problèmes de magnétophone

Les problèmes rencontrés au sujet du magnétophone à cassettes sont si fréquents que nous ne pouvions moins faire que de leur consacrer au moins une annexe.

Pour ce faire, et afin d'éviter une entrée prématurée dans l'asile psychiatrique le plus proche, il est bon d'adopter une certaine méthodologie.

Voici la marche à suivre pour vérifier le réglage de votre lecteur/enregistreur de cassettes (nous supposons correct le branchement des 3 câbles reliés à cet appareil).

Ecrire tout d'abord un petit programme de quelques lignes.
Par exemple :

```
10 CLS
20 PRINT "LE MAMA...LE MAGNET...LE MAGNETOPHONE"
30 PRINT "MAIS C'EST TRES SIMPLE...!"
```

Mettre ensuite une cassette dans l'appareil, en veillant à ce que la jonction amorce/bande magnétique soit placée au début de la cassette, approximativement face à la tête d'écriture.

Appuyez ensuite sur la touche rouge "RECORD" du magnétophone et, en maintenant cet appui, pressez la touche "PLAY".

Lâchez ensuite les deux touches, qui doivent alors rester dans leur position. *La cassette ne doit pas encore avancer.*

Réglez maintenant le volume à la valeur 6. Si vous avez un contrôle de tonalité, il doit être sur les aigus (HIGH). En fait, ces deux dernières instructions ne sont pas utiles durant la phase d'écriture, mais mieux vaut le faire tout de suite.

Frappez ensuite au clavier du TRS-80 la commande d'écriture du programme, suivie de la touche Enter :

CSAVE"A"

Dès ce moment, la cassette doit commencer à tourner, et le programme est supposé s'y inscrire. Lorsque la cassette s'arrête, le message READY doit être affiché sur l'écran.

Rebobinez alors la cassette, en pressant d'abord la touche STOP du magnétophone afin de faire remonter les deux touches enfoncées précédemment, puis la touche Rewind (rebobinage). La cassette doit alors tourner jusqu'à sa butée. Si ce n'est pas le cas, débranchez le cordon "Remote" du magnétophone (ancien modèle seulement).

Pressez ensuite la touche "Play" de l'appareil, et frappez au clavier :

CLOAD? (suivi de la touche "Enter")

La cassette doit alors commencer à tourner et, quelques secondes après, deux caractères astérisques (**) doivent apparaître dans le coin haut droit de l'écran. Cela indique que le début du programme a été reconnu par l'ordinateur (ce n'est que le début).

Si vous n'avez pas cette indication, inutile d'aller plus loin. Pressez le bouton "RESET" situé à l'arrière du clavier, retouchez légèrement le volume, et retournez au rebobinage.

Après plusieurs tentatives infructueuses, faites un nouvel essai d'écriture, car, en fait, l'erreur a pu se produire à ce moment-là. Eventuellement, changez de cassette, la vôtre peut avoir un défaut.

Si, contre toute attente, l'un des astérisques s'est mis à clignoter, c'est bon signe : le programme est bien lu, et compare avec son frère jumeau toujours en mémoire. Si tout se passe bien, le message READY ne doit pas tarder à apparaître sur l'écran.

Lorsqu'une erreur de comparaison est détectée, le message 'BAD' est affiché. Deux causes sont possibles : le programme a été mal écrit et, pour en avoir le coeur net, il faut tout recommencer ; ou bien il a été mal lu. Dans ce dernier cas, il faut retoucher légèrement le volume (essayez lentement et systématiquement toutes les positions), et relire le programme.

QUELQUES CONSEILS SUPPLEMENTAIRES :

Lorsque vous lisez un programme par la commande CLOAD, prenez l'habitude de faire suivre cette lecture d'une liste (LIST) afin de voir ce qui a été lu. En effet, il n'y a pas de contrôle de validité des informations avec cette commande. Dans le cas où une erreur s'est manifestée, il est bien rare que tout le reste du programme n'ait pas été "décalé". Dans ce cas, l'écran doit afficher un tas de choses incohérentes, et il faut refaire une lecture en retouchant légèrement le réglage de volume, en plus ou en moins.

Assurez-vous toujours, dans le cas où vous avez plusieurs programmes sur la même bande, de toujours commencer la lecture en début de programme (et non pas au milieu, même s'il s'agit d'un programme devant être sauté).

Normalement, lorsque vous avez déterminé la position correcte du réglage de volume, il ne faut plus y toucher. Toutefois, certains programmes du commerce demandent un réglage différent. Dans ce cas, remettez l'ancienne graduation, dès que la lecture est terminée.

A N N E X E V I

Le BASIC niveau I

Liste des instructions du Basic niveau I

PRINT

Imprime les valeurs de variables et d'expressions, ainsi que tout texte entre guillemets.

Exemple :

```
1Ø PRINT "A+B";A+B
```

INPUT

Commande à l'ordinateur de s'arrêter et d'attendre que vous entriez des données au clavier.

Exemple :

```
1Ø INPUT A,B,C
```

Peut aussi imprimer un texte avant de s'arrêter.

Exemple :

```
2Ø INPUT "ENTREZ PRIX";P
```

READ

Lit des valeurs contenues dans des instructions DATA

Exemple :

```
3Ø READ A,B,C,A$
```

DATA

Contient des données qui pourront être lues par des READ.

Exemple :

```
4Ø DATA 7,2,3,"BOULON"
```

RESTORE

Commande à la prochaine instruction READ de recommencer à partir de la première ligne DATA.

Exemple :

```
5Ø RESTORE
```

LET

Affecte une nouvelle valeur à la variable apparaissant à gauche du signe = (facultatif).

Exemple :

6Ø LET P=3.14159

GOTO

L'exécution se poursuit automatiquement à la ligne spécifiée.

Exemple :

7Ø GOTO 15Ø

IF-THEN

Correspond à un GOTO soumis à une condition.

Exemple :

8Ø IF A=B THEN 6Ø

IF

Forme plus générale : si la condition est vraie, les instructions de la ligne sont exécutées.

Exemple :

9Ø IF C=1 A=A+1:GOTO 1Ø

FOR-NEXT

Définit une boucle qui doit être exécutée un nombre déterminé de fois.

Exemple :

1Ø FOR I=1 TO 15
2Ø NEXT I

STEP

Spécifie la valeur de l'incrément dans une boucle FOR-NEXT.

Exemple :

3Ø FOR J=1 TO 5Ø STEP K

STOP

Arrête l'exécution du programme et affiche le message BREAK AT

###

Exemple :

4Ø IF A=B STOP

END

Arrête définitivement l'exécution du programme.

Exemple :

5Ø END

GOSUB

Commande l'exécution de la sous-routine commençant à la ligne spécifiée.

Exemple :

```
60 GOSUB 1000
```

RETURN

Termine l'exécution de la sous-routine ; l'exécution reprend à l'instruction qui suit le GOSUB.

Exemple :

```
70 RETURN
```

ON

Branchement à plusieurs voies dans un GOTO ou un GOSUB.

Exemple :

```
80 ON N GOTO 30,40,50
90 ON M GOSUB 100,200,300
```

Paramètres d'impression

AT

Spécifié après PRINT, commence l'impression à la position indiquée de l'écran.

Exemple :

```
10 PRINT AT 650,"SALUT"
20 PRINT AT X+1,A
```

TAB

Spécifié après PRINT, fait écrire la valeur qui suit après le nombre indiqué de positions, depuis le début de la ligne.

Exemple :

```
30 PRINT TAB(5);A;
TAB(I);B
```

Instructions graphiques

SET

Allume le segment spécifié sur l'écran.

Exemple :

```
10 SET(45,I+1)
```

RESET

Eteint le segment spécifié sur l'écran.

Exemple :

2Ø RESET (X,Y)

POINT

Prend la valeur 1 si le segment spécifié est allumé sur l'écran, et prend la valeur 0 sinon.

Exemple :

3Ø IF POINT(X,Y)=1 THEN PRINT "ALLUME"

CLS

Efface complètement l'écran.

Exemple :

4Ø CLS

Liste des commandes du Basic niveau I

NEW

Efface toutes les lignes de programme dans la mémoire.

Exemple :

NEW

RUN

Commande l'exécution du programme à partir de la ligne de numéro le plus petit.

Exemple :

RUN

RUN ###

Commande l'exécution du programme à partir de la ligne spécifiée.

Exemple :

RUN 3ØØ

LIST

Affiche les douze premières lignes du programme qui est en mémoire, à partir du numéro le plus petit ; pour afficher les lignes suivantes, utilisez la touche↑.

Exemple :

LIST

LIST ###

Même fonction que LIST, mais à partir de la ligne spécifiée.

Exemple :

LIST 355

CONT

Commande la poursuite de l'exécution du programme lorsque l'écran indique BREAK AT ~~###~~.

Exemple :

CONT

Remarque concernant ces commandes : Elles ne peuvent pas être incluses dans un programme.

ANNEXE VII

Conversions hexadécimal-décimal

Méthode et tables

Conversion hexadécimal-décimal

Convertir $1F8A_{16}$ en décimal. En lisant de gauche à droite on obtient :

$$\begin{array}{rcl}
 1F8A_{16} & \longrightarrow & A_{16} \times 10^0_{16} = A_{16} = 10_{10} \\
 & \longrightarrow & 8_{16} \times 10^1_{16} = 80_{16} = 128_{10} \\
 & \longrightarrow & F_{16} \times 10^2_{16} = F00_{16} = 3840_{10} \\
 & \longrightarrow & 1_{16} \times 10^3_{16} = 1000_{16} = 4096_{10} \\
 & & \hline
 1F8A_{16} & & 8074_{10}
 \end{array}$$

Conversion décimal-hexadécimal

Convertir $40\,003_{10}$ en hexadécimal. Dans la table on trouve :

$$\begin{array}{rcl}
 36\,864_{10} & = & 9_{16} \times 10^3_{16} = 9000_{16} \text{ et il reste } 3139_{10} \\
 3\,072_{10} & = & C_{16} \times 10^2_{16} = C00_{16} \text{ et il reste } 67_{10} \\
 64_{10} & = & 4_{16} \times 10^1_{16} = 40_{16} \text{ et il reste } 3_{10} \\
 3_{10} & = & 3_{16} \times 10^0_{16} = 3_{16} \text{ et il reste } 0 \\
 \hline
 40\,003_{10} & = & 9C43_{16}
 \end{array}$$

Hexadécimal	$\times 10^7_{16}$	$\times 10^6_{16}$	$\times 10^5_{16}$	$\times 10^4_{16}$	$\times 10^3_{16}$	$\times 10^2_{16}$	$\times 10^1_{16}$	$\times 10^0_{16}$
1	268 435 456	16 777 216	1 048 576	65 536	4 096	256	16	1
2	536 870 912	33 554 432	2 097 152	131 072	8 192	512	32	2
3	805 305 368	50 331 648	3 145 728	196 608	12 288	768	48	3
4	1 073 741 824	67 108 864	4 194 304	262 144	16 384	1 024	64	4
5	1 342 177 280	83 886 080	5 242 880	327 680	20 480	1 280	80	5
6	1 610 612 736	100 663 296	6 291 456	393 216	24 576	1 536	96	6
7	1 879 048 192	117 440 512	7 340 032	458 752	28 672	1 792	112	7
8	2 147 483 648	134 217 728	8 388 608	524 288	32 768	2 048	128	8
9	2 415 919 104	150 994 944	9 437 184	589 824	36 864	2 304	144	9
A	2 684 354 560	167 772 160	10 485 760	655 360	40 960	2 560	160	10
B	2 952 790 016	184 549 376	11 534 336	720 896	45 056	2 816	176	11
C	3 221 225 472	201 326 592	12 582 912	786 432	49 152	3 072	192	12
D	3 489 660 928	218 103 808	13 631 488	851 968	53 248	3 328	208	13
E	3 758 096 384	234 881 024	14 680 064	917 504	57 344	3 584	224	14
F	4 026 531 840	251 658 240	15 728 640	983 040	61 440	3 840	240	15

* Ces documents sont reproduits avec l'aimable autorisation de la Compagnie CII HONEYWELL BULL.

Arithmétique hexadécimale :

Table d'addition

0	1 2 3	4 5 6	7 8 9	A B C	D E F
1	02 03 04	05 06 07	08 09 0A	0B 0C 0D	0E 0F 10
2	03 04 05	06 07 08	09 0A 0B	0C 0D 0E	0F 10 11
3	04 05 06	07 08 09	0A 0B 0C	0D 0E 0F	10 11 12
4	05 06 07	08 09 0A	0B 0C 0D	0E 0F 10	11 12 13
5	06 07 08	09 0A 0B	0C 0D 0E	0F 10 11	12 13 14
6	07 08 09	0A 0B 0C	0D 0E 0F	10 11 12	13 14 15
7	08 09 0A	0B 0C 0D	0E 0F 10	11 12 13	14 15 16
8	09 0A 0B	0C 0D 0E	0F 10 11	12 13 14	15 16 17
9	0A 0B 0C	0D 0E 0F	10 11 12	13 14 15	16 17 18
A	0B 0C 0D	0E 0F 10	11 12 13	14 15 16	17 18 19
B	0C 0D 0E	0F 10 11	12 13 14	15 16 17	18 19 1A
C	0D 0E 0F	10 11 12	13 14 15	16 17 18	19 1A 1B
D	0E 0F 10	11 12 13	14 15 16	17 18 19	1A 1B 1C
E	0F 10 11	12 13 14	15 16 17	18 19 1A	1B 1C 1D
F	10 11 12	13 14 15	16 17 18	19 1A 1B	1C 1D 1E

Table de multiplication

1	2 3	4 5 6	7 8 9	A B C	D E F
2	04 06	08 0A 0C	0E 10 12	14 16 18	1A 1C 1E
3	06 09	0C 0F 12	15 18 1B	1E 21 24	27 2A 2D
4	08 0C	10 14 18	1C 20 24	28 2C 30	34 38 3C
5	0A 0F	14 19 1E	23 28 2D	32 37 3C	41 46 4B
6	0C 12	18 1E 24	2A 30 36	3C 42 48	4E 54 5A
7	0E 15	1C 23 2A	31 38 3F	46 4D 54	5B 62 69
8	10 18	20 28 30	38 40 48	50 58 60	68 70 78
9	12 1B	24 2D 36	3F 48 51	5A 63 6C	75 7E 87
A	14 1E	28 32 3C	46 50 5A	64 6E 78	82 8C 96
B	16 21	2C 37 42	4D 58 63	6E 79 84	8F 9A A5
C	18 24	30 3C 48	54 60 6C	78 84 90	9C A8 B4
D	1A 27	34 41 4E	5B 68 75	82 8F 9C	A9 B6 C3
E	1C 2A	38 46 54	62 70 7E	8C 9A A8	B6 C4 D2
F	1E 2D	3C 4B 5A	69 78 87	96 A5 B4	C3 D2 E1

Conversion hexadécimal-décimal de nombres entiers

La table située pages suivantes permet d'effectuer des conversions entre :

- les entiers hexadécimaux compris entre 0 et FFF d'une part, et
- les entiers décimaux compris entre 0 et 4095 d'autre part.

Pour convertir des entiers plus grands, les valeurs trouvées dans la table doivent être ajoutées aux nombres du tableau ci-dessous.

Hexadécimal	Décimal	Hexadécimal	Décimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

LA PRATIQUE DU TRS-80

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0199	0198	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0215	0214	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0231	0230	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0247	0246	0248	0249	0250	0251	0252	0253	0254	0255
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

Achevé d'imprimer
sur les presses de
Impressions Cepi
174, Quai de Jemmapes
75010 PARIS
Dépot légal N ° 58 - 3e trimestre 1979

N ° d'édition : 86470-5-1
ISBN : 2-86470-005-0

La pratique du TRS-80

Volume I - Architecture et Basic II

Cet ouvrage de la série perfectionnement s'adresse aux amateurs de TRS-80 ayant déjà quelques connaissances en informatique et des notions de BASIC. Il a pour objectif de permettre au possesseur ou utilisateur de TRS-80 de tirer le meilleur parti de son ordinateur — et — en particulier du langage BASIC niveau II. Il comprend des exemples de programmes et des annexes pratiques.

Editions du P.S.I.
B.P. 86
77400 Lagny/Marne